R + EXCEL

Prokaj Vilmos
prokaj@cs.elte.hu

TEXelés időpontja: 2015. szeptember 5.

Verziószámok

Rxls:	2.0.292
com:	1.0.80
comproxy:	1.0.26

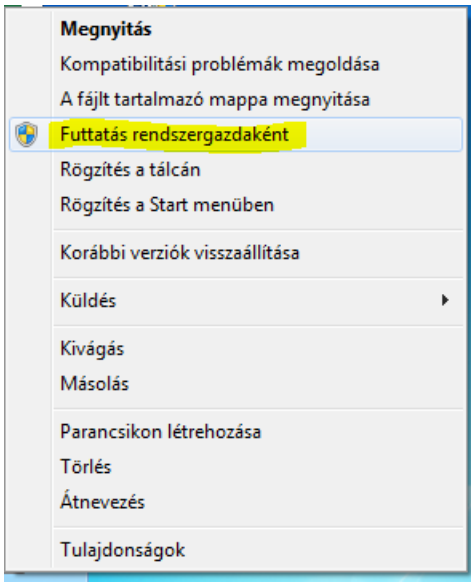
Tartalomjegyzék

1 Türelmetleneknek

1.1. Installálás

A mellékelt fileok között megtalálhatóak a `comproxy`, `com` ill. `Rxls` csomagok win-binary formátumban. A telepítést ebben a sorrendben kell elvégezni, pl. a menü használatával. A `comproxy` és `com` csomagok telepítését rendszegazdaként célszerű elvégezni, és rögtön a telepítés után érdemes regisztrálni az `R com` szerverét.

A regisztráció csak akkor sikerül, ha rendszergazda jogosultsággal hajtjuk végre. Pontosabban a `com` csomag regisztrációjának elvégzéséhez, az `R` programot az ikon felett a jobb egérgombra klikkelve, a `Futtatás rendszergazdaként` lehetőséget kiválasztva kell elindítani, lásd az 1.1 ábrán.



1.1. ábra. R indítása rendszergazdaként.

A `comCheckRegistry` függvény ellenőrzi a registry bejegyzést, lásd a 10.1 szakaszt.

```
> comRegisterRegistry()
NULL
> comCheckRegistry()
[1] TRUE
```

Az `Rxls` csomag telepítéséhez nem szükséges rendszergazdai jogosultság. Miután ennek telepítése is megtörtént, be kell tölteni az `Rxls` csomagot

```
> require(Rxls)
Loading required package: Rxls
Loading required package: com
Loading required package: comproxy
Az Rxls csomag telepítése hiányos vagy sérült.
Futassa le az 'installRxls()' parancsot!
> installRxls()
C:\Users\prokaj\AppData\Roaming/Rxls könyvtár létrehozása.
Másolandó fileok: Calc.xlt, Calc.xltm, Rdev.xlam, R.xls
A másolás sikeresen megtörtént.
Verzió információ frissítése...ok
```

A betöltés során valószínűleg kapunk egy figyelmeztetést, hogy az `installRxls()` parancsot érdemes lefuttatni. Ez egyrészt létrehozza az `{APPDATA}/Rxls` könyvtárat és ide másolja a `R.xls`, `Rdev.xlam` és `Calc.xlt` munkafüzeteket valamint létrehozza a verzió információt tartalmazó `versions` nevű file-t. Alternatív lehetőség a mellékelt `R-COM_setup_x.x-x.exe` legfrisseb változatának használata.

1.2. EXCEL használata R-ből és fordítva

Tegyük fel, hogy az `Rxls` csomag telepítése sikeresen megtörtént, az `installRxls` hiba nélkül lefutott. Elsőként az `Rdev.xlam` munkafüzetet érdemes megnyitni. Ha rendszeresen használjuk bővítményként is telepíthetjük, ekkor nem kell mindig megkeresni és betölteni.

Az `Rdev.xlam` munkafüzet az alábbi címen lálható

```
<USER>/<APPDATA>/Rxls/Rdev.xlam
```

Alapértelmezésben az `{APPDATA}` könyvtár rejtett. Ezt a vezérlő pult használatával felfedhetjük (rejtett mappák megjelenítése). Másik lehetőség, hogy a windows intéző címsorába beírjuk az elérési utat. R-ből a pontos elérési utat az appdata környezeti változóból is kiolvashatjuk:

```
> Sys.getenv("appdata")
[1] "C:\\Users\\prokaj\\AppData\\Roaming"
```

Ez egyben az első tesztje annak, hogy az elképzelt mechanizmus működik-e más gépeken is. Az `Rdev.xlam` megnyitásakor az EXCEL megpróbálja megnyitni a hivatkozott `R.xls` munkafüzetet is. Egyáltalán nem nyivánvaló, hogy ez sikerül is, ugyanis az EXCEL a teljes elérési utat megjegyzi, ami nyilván nem létezik, ha a munkafüzetet egy másik gépre visszük át. Ezért a `Workbook_open` rutin ellenőrzi, hogy létezik-e megnyitott `R.xls` munkafüzet. Ha nincs ilyen nevű munkafüzet megnyitva, akkor megpróbálja megnyitni az `{appdata}/Rxls` könyvtárból. Ha telepítés rendben megtörtént, akkor ez sikerül is. Ha minden rendben ment, akkor az újabb, ribbon-t használó EXCEL változatok esetén a Bővítmények fül alatt, régebbi EXCEL változatok esetében pedig a menüsorban megjelenik két almenü, az egyik felirata R, a másiké `Rxls development`. A telepítés legegyszerűbb tesztelése, ha ráklikellünk az

```
R->R ablak megjelenítés
```

menüpontra. Hatására az EXCEL megnézi, hogy van-e futó R alkalmazás, ha van akkor az láthatóvá válik, ha nincs akkor elindít egy új R példányt. A következő lépés az R és EXCEL összekötése az

`R -> R<->EXCEL összekötés`

menüponttal. Ez az EXCEL új Ribbon kezelő felületén a Bövítmények fül alatt van, de elérhető az Rxls development fülön is. Ennek hatására az R oldalon létrejön egy változó THISXL névvel. Ez reprezentálja az EXCEL alkalmazást. Pontosabban ez a változó a .XL<azonosító> nevű környezeten belül jelenik meg. Ahhoz, hogy az Rxls csomag rutinjai is „lássák" a keresési útvonalhoz is csatoljuk. Ennek a megoldásnak az az előnye, hogy a kapcsolat bontásakor (azaz az EXCEL bezárásakor egyszerűbben lehet az adott EXCEL példányra történő hivatkozásokat törölni).

```
> ls.str(envir = .XL393694, all.names = TRUE)
THISXL : Class 'COMObject' <externalptr>
wb_Munkafüzet1 : <environment: 0x155198c0>
> ls.str(envir = .XL393694$wb_Munkafüzet1, all.names = TRUE)
.wbname :  chr "Munkafüzet1"
.wsname :  chr "Munka1"
> ls.str(2,all=TRUE)
.wbname :  chr "Munkafüzet1"
.wsname :  chr "Munka1"
THISXL : Class 'COMObject' <externalptr>
> search()[1:5]
[1] ".GlobalEnv"
[2] ".XL393694:wb_Munkafüzet1"
[3] "package:Rxls"
[4] "package:com"
[5] "package:comproxy"
```

Ennek segítségével ugyanúgy használhatjuk a hivatkozott EXCEL példányt R-ből, mint Visual Basic-ből. Például, létrehozhatunk egy új munkafüzetet és abban az $A$1 cellát kitölthetjük értékkel, vagy formulával. A formátum első ránézésre talán szokatlan. Ahol a Visual Basic kódban pont van, ott vagy egy tulajdonságot kérdezünk le, vagy egy metódust alkalmazunk. A com csomag korábbi változataiban a két esetnek eltérő a szintakszisa: tulajdonságot

`obj[[<tulajdonságnév>, <argumentumok>]]`,

míg metódust

`obj$<metódusnév>(<argumentumok>)`

alakban lehet elérni. Az új változatban, 1.0-52-től felfelé, lehetőség van a `<obj>$<név>()` szintakszis használatára tulajdonság lekérdezés esetén is, lásd a 10.1 szakaszt.

```
> wb<-THISXL[["workbooks"]]$add()
> cell<-wb[["sheets",1]][["range","$A$1"]]
> cell[["value"]]
[1] NA
> cell[["value"]]<-1
> cell[["value"]]
[1] 1
> cell<-cell$offset(1,0)
> cell[["formula"]]<-"=$A$1+2"
> cell[["formula"]]
[1] "=$A$1+2"
```

```
> cell[["value"]]
[1] 3
> cell<-NULL
```

Az Rxls csomag célja, hogy a gyakran elő forduló adatmozgatásokat egyszerűen lehessen végrehajtani. Pl. adott EXCEL tartomány értékét az XLget függvénnyel is elérhetjük. Az XLreaddf és XLwritedf függvények data.frame-t olvasnak be ill. írnak EXCEL tartományból, ill. tartományba.

```
> df <- data.frame(Dátum = seq.Date(as.Date("2012-01-01"),
+ by = "1 month", length = 5))
> df$díj <- rnorm(nrow(df), 100)
> df$kárkifizetés <- rnorm(nrow(df), 70)
> df
       Dátum       díj kárkifizetés
1 2012-01-01  98.92022     69.17676
2 2012-02-01 100.15791     71.55648
3 2012-03-01 101.11234     69.50647
4 2012-04-01 100.74544     69.49601
5 2012-05-01  99.09140     69.50164
> XLwritedf(XLrange = THISXL[["range", "$C$1"]], df = df,
+ with.names = TRUE, autoFit = TRUE, setname = "havi_bontás")
NULL
```

Az utolsó sor hatására az EXCEL aktív munkalapjára másolódik a df data.frame a változónevekkel együtt (ez a táblázat felső sora lesz), az oszlop szélességeket az éppen másolt táblázathoz illesztjük, végül létrehozunk egy nevet az aktív EXCEL munkafüzetben, ami arra a tartományra hivatkozik, ahová az adatokat másoltuk.

```
> ## wb a korábban létrehozott új munkafüzet
> address<-wb[["names"]]$item("havi_bontás")[["refersto"]] #$
> address
[1] "=Munka1!$C$1:$E$6"
> df1<-XLreaddf.cols(address)
> df1
  Dátum       díj kárkifizetés
1 40909  98.92022     69.17676
2 40940 100.15791     71.55648
3 40969 101.11234     69.50647
4 41000 100.74544     69.49601
5 41030  99.09140     69.50164
> ## vagy
> range<-wb[["names"]]$item("havi_bontás")[["referstorange"]]
> range$address(external=TRUE)
[1] "[Munkafüzet2]Munka1!$C$1:$E$6"
> df2<-XLreaddf.cols(XLrange=range)
> identical(df1,df2)
[1] TRUE
```

Látható, hogy a dátum adatok részben elvesztek. Ennek az az oka, hogy az XLreaddf.cols ill. XLreaddf.rows függvények a tartomány value2 tulajdonságát használják. Amikor ezek

a rutinok születtek a `com` (lánykori nevén `rcom`) nem tudta kezelni a dátum típusú adatokat. Kerülő megoldásként született az `XLDate` függvény, ami a numerikus adatot visszaalakítja `Date` típusúvá.

```
> XLDate(df1$Dátum)
[1] "2012-01-01" "2012-02-01" "2012-03-01" "2012-04-01"
[5] "2012-05-01"
```

1.3. Egyszerű minta projekt

Tegyük fel, hogy van egy `R` scriptünk `calc.R` névvel.

```
> cat(readLines("calc.R"), sep = "\n")
calc.cn<-function(lx,
                  nu=1/(1+i),
                  i=if (!missing (nu)) (1/nu)-1){
  x <- seq_along(lx)-1
  ## 0-val indul az indexelés
  lx1<- lx[-length(lx)]
  qx <- -0.95*diff(lx)/(lx1+(lx1==0))
  ## módosított halandóság
  lx <- lx[1]*cumprod(c(1,1-qx))
  ## módosított l
  df <- data.frame(x=x,lx=lx,qx=c(qx,1),Dx=lx*nu^x)
  df$dx <- with(df,lx*qx)
  df$Cx <- with(df,dx*nu^(x+.5))
  df$Nx <- rev(cumsum(rev(df$Dx)))
  df$Mx <- rev(cumsum(rev(df$Cx)))
  df$Rx <- rev(cumsum(rev(df$Mx)))
  attr(df,"nu")<-nu
  attr(df,"i")<-i
  df
}
## teszt adat
lx<-seq(1e5,0,length.out=101)
cn<-calc.cn(lx,i=.05)
```

Ez a script kommutációs számokat számol módosított halandósági táblából, a halálozási rátát módosítjuk az eredeti 95 %-ára. A `calc.cn` függvénynek három argumentuma van, ebből legalább kettőt kell megadni, az `lx` halandósági függvényt és a `nu` diszkont ráta ill. `i` technikai kamat közül az egyiket, vagy mindkettőt. Ez a számolás csak a diszkont rátát használja, de az ereményben megjegyezzük a technikai kamatot `i`-t is.

```
> source("calc.R", local = TRUE)
> str(cn)
'data.frame': 101 obs. of  9 variables:
 $ x : num  0 1 2 3 4 5 6 7 8 9 ...
 $ lx: num  100000 99050 98100 97149 96197 ...
 $ qx: num  0.0095 0.0096 0.00969 0.00979 0.0099 ...
 $ Dx: num  100000 94333 88979 83921 79142 ...
```

```
 $ dx: num  950 950 951 951 952 ...
 $ Cx: num  927 883 842 802 764 ...
 $ Nx: num  1698778 1598778 1504445 1415466 1331545 ...
 $ Mx: num  19578 18651 17767 16925 16123 ...
 $ Rx: num  402395 382817 364167 346400 329474 ...
 - attr(*, "nu")= num 0.952
 - attr(*, "i")= num 0.05
```

Tegyük fel, hogy a bemenő adatokat `lx`, `nu` és `i`-t szeretnénk `EXCEL`-ben beállítani és az eredményül kapott táblázatot a számoló munkafüzet új lapjára szeretnénk írni. A `lx` halandósági függvény az `EXCEL` három oszlopos táblázata, kor, férfi és női oszlopokkal. Azt szeretnénk, hogy a számoló munkafüzetben be lehessen állítani, a technikai kamat értékét, azt, hogy férfi, vagy női haladósággal akarunk számolni, valamint azt is, hogy mi legyen az új munkalap neve.

A megvalósításhoz nyissuk meg az `Rdev.xlam` munkafüzetet. Ha nincs beállítva, akkor engedélyezzük a `Visual Basic` projektekhez való hozzáférést. Hozzunk létre egy új számoló munkafüzetet az `Rxls development` menü `Új számoló munkafüzet` parancsával. A felugró mentés ablakban adjuk meg az új munkafüzet nevét, alapértelmezésben `Calc1.xls`. A fileformátum lehet `xlsm` is. Az új munkafüzetben két lap van `Adatok` ill. `R kód` névvel. Első lépésben az `Adatok` munkalapot töltsük fel. A jobb egérgomb alatti menüben az `Insert Form` pont alatt található néhány gyakrabban előforduló elem.

Jelöljük ki a `$B$5:$B$7` tartományt és a jobb egérgomb alatt felnyíló menüt használva illesszünk be egy `Data Line`-t. A `$B` oszlopban cseréljük le az `Adat` neveket a következőkkel: `Technikai kamat`, `Nem` és `Eredmény helye`. Töltsük ki az `# R nevek #` oszlopot `i`, `nem`, `wsname`-mel. Ezek lesznek a változó nevek az `R` oldalon. Ezután adjuk meg az alapértelmezett értékeket: a technikai kamat esetén ez lehet pl. 0.02 a nem esetén férfi, míg az Eremény helye mező esetén lehet egy képlet, ami a technikai kamat és a nem értékéből kiszámolja az új munkalap nevét, pl.

```
="Komm. számok("&$D$6&",i="&$D$5&")"
```

Végül mivel a `Nem` mező esetén csak két lehetséges értékből lehet választani, töltsük ki a `$C$6` mezőt `férfi;női` értékkel. Ez a választható értékek felsorolása pontosvesszővel elválasztva. Ha most kijelöljük a `$C$6` mezőt és a jobb egérgomb alatt felnyíló menü segítségével beszúrunk egy `Drop Down Data`-t, akkor az `R` kód munkalap `$A` oszlopának aljára másolódik a lehetséges értékek listája, és a `$C$6` cellában megjelenik egy lenyíló vezérlő elem. A vezérlő elem alatt a `$C$6` cella zárolt lesz, és a képletezés hatására mindig a választott értéket tartalmazza.

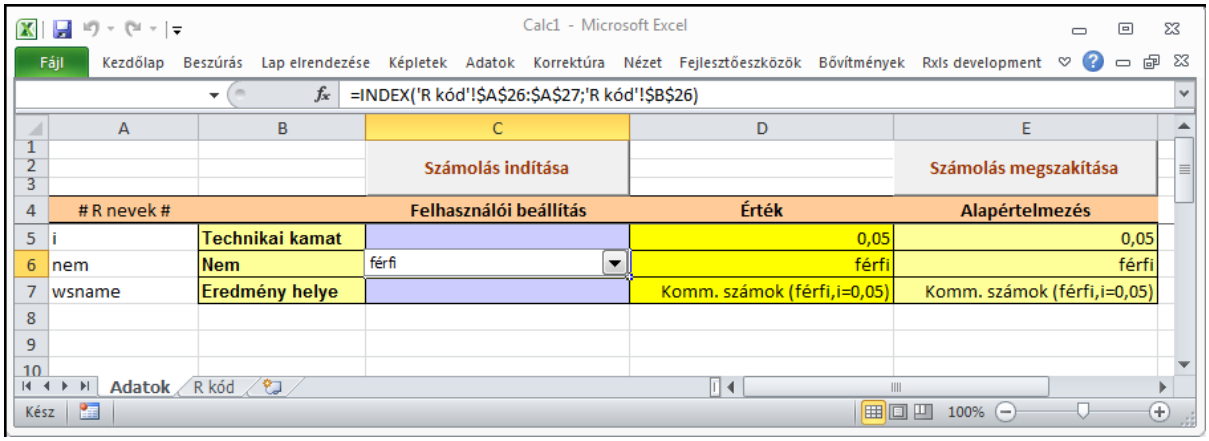
Az így kialakított munkalap képe:

Ha ezen a ponton a `Számolás indítása` gombra klikkelünk, akkor az `Adatok` munkalapon definiált változók megjelennek az `R` oldalon is. Ezt ellenőrizhetjük az `ls.str()` parancs beírásával az `R` konzolba. A `R` ablakot az `EXCEL`

```
Bővítmények -> R -> R ablak megjelenítése
```

menüponttal jeleníthetjük meg, ha az háttérben van.

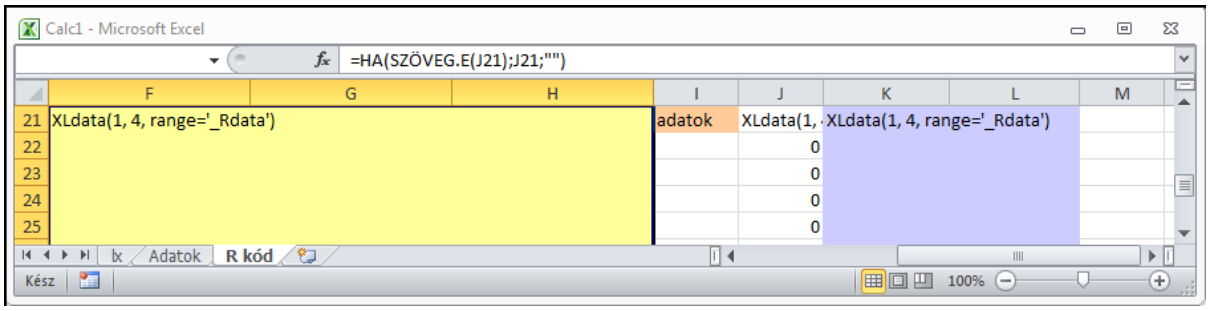
```
> ls.str(env = .XL393694$wb_Calc1, all.names = TRUE)
# R nevek # :  chr "Érték"
.FUN :   expression({     XLdata(1, 4, range = "_Rdata") })
.wbname :  chr "Calc1"
```



1.2. ábra. `Adatok` munkalap

```
.wsname :  chr "Adatok"
i :  num 0.05
nem :  chr "férfi"
wsname :  chr "Komm. számok (férfi,i=0,05)"
```

A `Számolás indítása` gomb megnyomásakor annyi történt, hogy az `EXCEL` az `R` oldalon létrehozott egy környezetet, ami az adott `EXCEL` példány és aktív munkafüzet adatait tartalmazza: hivatkozás az `EXCEL.Application`-ra és a munkafüzet neve, és ebben a környezetben végrehajtotta a `R kód` munkalapon összerakott kódot. Ez ezen a ponton egyetlen utasítást tartalmaz, ami a `_Rdata` tartomány (ez az `Adatok` munkalap első négy oszlopa) első oszlopából a változó neveket, a 4. oszlopából az értékeket veszi.



1.3. ábra. Az `'R kód'` munkalap részlete

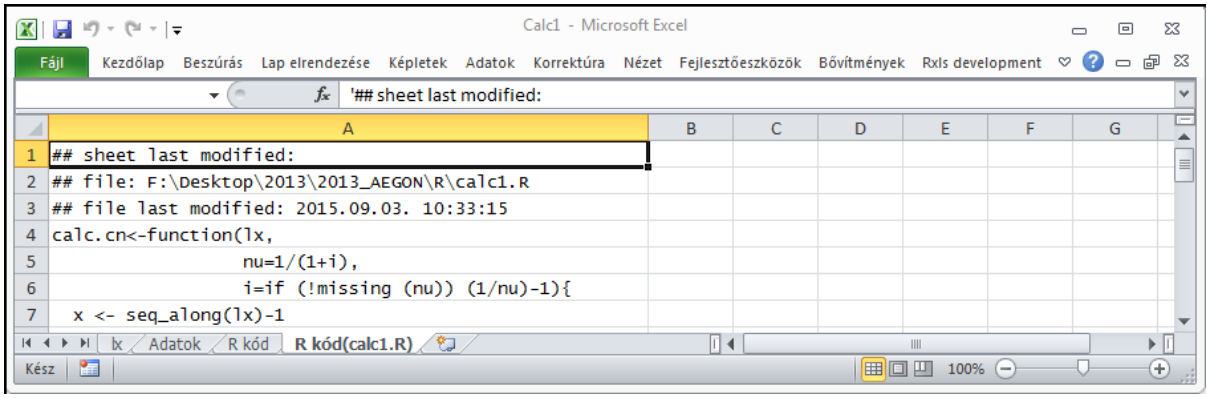
Ezzel persze még a számolást nem tudjuk elvégezni, mert nem adtuk át a halandósági függvényt `lx`-et, amire szintén szükség van a számoláshoz. Ha ezt is az `EXCEL` munkafüzetben szeretnénk tárolni, akkor illesszük be egy új munkalapra. Az `R` kódot úgy fogjuk módosítani, hogy a halandósági táblát a munkafüzet `lxdata` nevű tartományából vegye. Ezért a halandósági táblát tartalmazó tartománynak adjuk meg ezt a nevet az `EXCEL` névkezelőjében.

Utolsó lépésként meg kell adnunk a `Calc1.xls` munkafüzetben azt is, hogy az `R`-nek átadott adatokkal milyen számolást is végezzen. Ha hordozhatóvá akarjuk tenni a munkafüzetet, akkor erre a legegyszerűbb megoldás, ha a számolást végző `R` kódot is a munkafüzetben tároljuk. Ezt `R kódlap` beszúrásával tehetjük meg. A `calc.R` közvetlenül nem alkalmas a céljainkra, a szkript végét módosítani kell, hogy a számolást az `EXCEL`-ből kapott adatok alapján végezze. A módosított és az eredeti `R` szkript is megtalálható a mellékelt file-ok között.

```
> calc.R <- readLines("calc.R")
> calc1.R <- readLines("calc1.R")
> cat("új sorok:", setdiff(calc1.R, calc.R), "\ra régi helyett:",
+                  setdiff(calc.R, calc1.R), sep = "\n")
új sorok:
withObj(THISXL$workbooks(.wbname)$worksheets(),{
          if(is.null(.$item(wsname)))
            with(.$add(),{
                   .[["name"]]<-wsname
                   lx<-XLreaddf.cols('lxdata')
                   cn<-calc.cn(lx[[nem]],i=i)
                   XLwritedf (cn,XLrange=.$cells(1,1))
                 })
      })

a régi helyett:
## teszt adat
lx<-seq(1e5,0,length.out=101)
cn<-calc.cn(lx,i=.05)
```

Az így módosított kódot szúrjuk be `R kódlap`-ként a munkafüzetbe. Az `R kódlap beszúrás` funkciót két helyen is megtalálhatjuk, egyrészt az `Rxls development` menüpont alatt, másrészt a munkalap fülekről felnyíló menüben is. A művelet eredményeként egy új munkalap jön létre `R kód(calc1.R)` névvel, lásd az 1.4 ábrát.



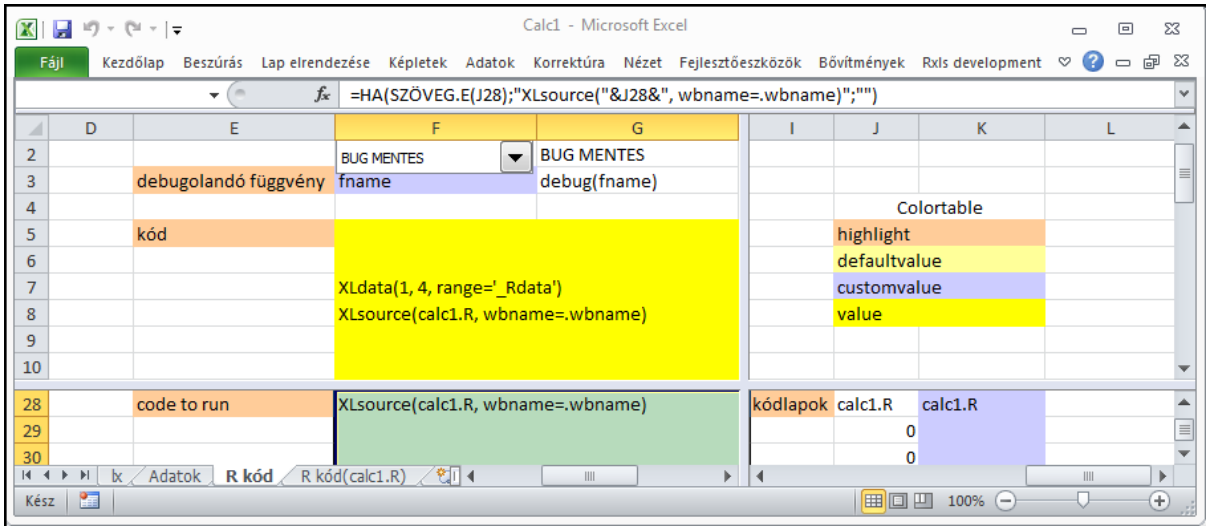
1.4. ábra. Az 'R kód(calc1.R)' kódlap

Ezután be kell jegyezni az `R kód` munkalapon, hogy a `calc1.R` kódlapot az R-nek végre kell hajtania. Ehhez az `R kód` munkalapon töltsük ki a kódlapok mező első sorát az 1.5 ábrán látható módon.

Ezzel a `Calc1.xls` munkafüzet elkészült. Utolsó lépésként kattintsunk a

`Bővítmények -> Rxls development -> Véglegesítés`

gombra. Ekkor az `R kód` nevű munkalapok eltűnnek, az `Adatok` munkalapon az első oszlop elrejtődik, és a látható munkalapok védetté válnak. Ezután csak a kékes színű felhasználói adatok részére fenntartott mezőkbe lehet írni.



1.5. ábra. Az ’R kód’ munkalap kitöltve

2 Részletesebben

Az elképzelt munkamenet a következő. Van egy R szkriptünk, aminek a paraméterezését egy EXCEL munkafüzetből szeretnénk megoldani.

2.1. Adatok

Első lépésként azonosítani kell a bemenő adatokat és el kell készíteni a számoló munkafüzet Adatok munkalapját. Ha nagyobb mennyiségű fix paramétert is át kell adni, akkor azokat egy külön munkalapon is elhelyezhetjük. Ha ennek, az extra adatokat tartalmazó, munkalapnak a neve R kód-dal kezdődik, vagy a (hidden)-nel végződik, akkor a véglegesítés során az adott munkalap rejtetté válik.

Az adatok átvitelére az XLdata függvényt célszerű használni. Ennek részletes leírását lásd a 7.1 szakaszban. Itt megelégszünk annyival, hogy XLdata (n1,n2,range="<range cím>") függvény hatására a <range cím> által meghatározott tartomány n1 oszlopában lévő értékeket változó névként, az n2 oszlopban mellette lévő értékeket a változó értékeként értelmezi az R. Ha a név mező üres vagy nem szöveges típusú, akkor az adott sorhoz nem tartozik változó definíció. Ha egy név többször is szerepel, akkor az utolsó sor változó definíciója érvényes. Az n1,n2 értékek megadása nem kötelező, ha viszont hiányzik, akkor a <range cím> sztringnek egy legalább két oszlopos tartományt kell kijelölnie és ennek első két oszlopát fogja használni a program. A <range cím> teljes oszlopokra is mutathat, pl $A:$C. Ebben az esetben csak a munkalap UsedRange tartományát használjuk.

Alapértelmezésben, csak a látható sorokat használjuk és egyesített cellák esetén a cellában látható értéket használjuk.

Abban az esetben, ha a változó név nem megengedett karaktereket is tartalmaz, pl. szóköz, ékezetes karakterek, számmal kezdődő név stb. akkor az R oldalon a változó nevét ‘ (back quote) karakterek közé kell tenni. Pl. az ‘# R nevek #‘ nevű változó értéke (ez az Adatok munkalap 5. sora) "Érték" lesz, ahogy ez a 9. oldalon látható.

2.2. Kód

Az R szkriptet célszerű két részre bontani: függvény definíciós részre és az aktuális számolást végző részre. Ennek csupán teszteléskor van jelentősége. Ha nem az történik, amit várunk, akkor kénytelenek vagyunk a hibát megkeresni a kódban. Az R kód munkalapon kijelölhetjük, hogy melyik függvényt működését akarjuk ellenőrizni. Ennek az az eredménye, hogy az adatok és a függvények beolvasása után a debug (<fv. név>) is végrehajtásra kerül és így a debugolásra kijelölt függvényt lépésenként lehet kiértékelni.

Célszerű a függvényeket és a számolást végző kódot, mondjuk R studio-ban, létrehozni és tesztelni. Ezután a kész kódot elmenteni R szkriptként (.R kiterjesztéssel) majd azt importálni a számoló munkafüzetbe R kódlapként. Így is elő fordulhatnak váratlan dolgok, de ezek nagy részét kiszűrhetjük, ha a kód megírása során a bemenő adatokat, vagy is a futási környezetet már az EXCEL adja. Ezalatt azt értem, hogy az első lépésként létrehozott Adatok munkalapot

kitöltjük a teszt paraméterekkel, összekötjük az R példányunkat az EXCEL-lel (Rxls menü R <-> EXCEL összekötés) és ráklikkelünk a Számolás indítása feliratú gombra. Mivel még nincs semmilyen végrehajtandó kód a számoló munkafüzetben, ennek hatására a teszt adatok, az EXCELben definiált változó nevekkel, átkerülnek az R-be. Ebben az R környezetben érdemes dolgozni.

3 Az EXCEL oldal felhasználói szemmel

Az Rxls R csomagban három EXCEL munkafüzet van.

R.xls Ez egy Office XP kompatibilis munkafüzet. Ez az amit az alkalmazások ténylegesen használnak. Ebben van megoldva a R példány megkeresése és az összeköttetés létrehozása, valamint néhány olyan rutin is ami a számoló munkafüzetekben szükséges lehet: file és mappa választás, valamint ACCESS tábla választás.

Rdev.xlam Ez a munkafüzet egy bővítmény, ami a számoló munkafüzetek létrehozásához ad segítséget.

Calc.xltm A számoló munkafüzetek templateje.

3.1. Az Rdev.xlam bővítmény

Rdev.xlam

Ebbe a munkafüzetbe gyűjtöttem azokat a rutinokat, melyek csak a fejlesztéshez szükségesek. Ezek munkalap védelem be és kikapcsolása, R kód elrejtése és felfedése, valamint a munkafüzet véglegesítése. A véglegesítés a munkalapok védelmének bekapcsolását, a kódot tartalmazó munkalapok elrejtését jelenti. Ennek során a védett munkalapokon a felhasználó által kitöltendő mezők zárolása feloldódik. Az, hogy melyik cella tartozik ebbe a csoportba a kitöltés színe határozza meg.

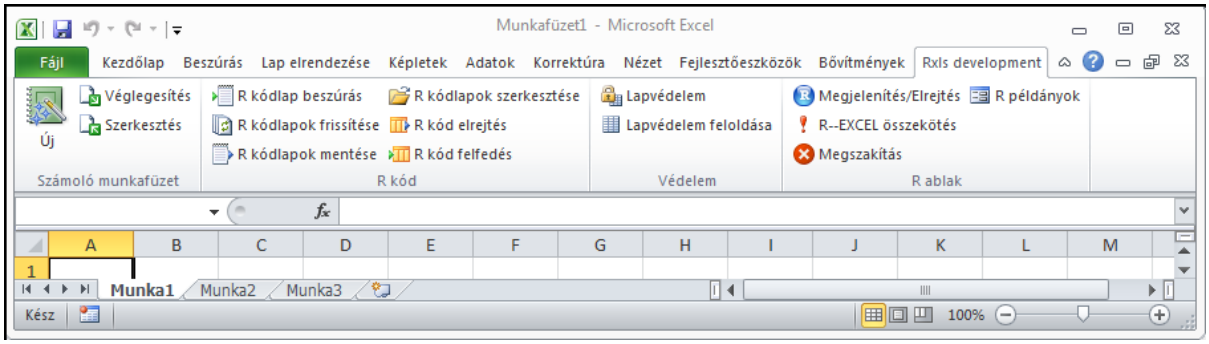
A tényleges számolást végző R rutinokat talán legegyszerűbb magában a számoló munkafüzetben tárolni. Másik lehetőség lehetne, hogy a feladatra szabott R csomagot készítünk és azt telepítjük a felhasználó gépére.

Az R kódot tehát EXCEL munkalapon szeretnénk tárolni. Szerkeszteni, tesztelni viszont nem itt célszerű. A követendő eljárás az lehet, hogy pl. Rstudio-ban fejlesztünk és a kész kódot bemásoljuk a munkafüzetbe. Az ilyen R kódot tartalmazó munkalapokat a továbbiakban R kódlapnak fogom nevezni.

Ez a bővítmény számos új menüpontot definiál. Az EXCEL újabb, RIBBON felületet használó, változataiban létrehoz egy Rxls development tabot, a korábbi változatokban pedig ugyanilyen névvel egy menüpontot, az EXCEL fő menüsorában. Emellett a cell menü is kiegészül néhány ponttal és a munkalap fülek menüje is kap egy új menüpontot.

Az Rxls development fül elemei

A Rxls development fülön lévő parancsok elérhetőek a Bővítmények fül alatti menüpontokból is. Az EXCEL korábbi, nem a RIBBON felületet használó változataiban pedig a menüsorban bővül ki két ponttal.



3.1. ábra. Az 'Rxls development' menü.

Számoló munkafüzet csoport

Itt csak három gomb található, Új, Véglegesítés ill. Szerkesztés névvel. Az Új gomb, létrehoz egy új számoló munkafüzetet. Ez a Calc.xltm template egy másolata lesz. Célszerű rögtön elmenteni a létrehozás után valamelyik makróbarát formátumba. Ennek oka, hogy néhány munkalapfüggvény (INFO, CELLA) másként viselkedik mentett munkafüzetben, mint mentetlenben.

A Véglegesítés menüparancs hatására a számoló munkafüzet R kódlapjai rejtetté válnak, hasonlóan az Adatok munkalap # R nevek # oszlopához. Valamennyi cella zárolt lesz kivéve azokat, amelyek színe megegyezik a customvalue színnel. Ez a szín R kód munkalap colortable tartományából derül ki. A tartomány elemei színes cellák, a cella értéke a szín neve. Végül a kapott munkafüzetet elmentjük.

A Szerkesztés a Véglegesítés ellentéte. A munkalapvédelmet kikapcsolja, a rejtett munkalapokat és oszlopokat láthatóvá teszi.

R kód csoport

Az első oszlopban az R kódlapok kezelésére szolgáló parancsok vannak. Beszúrás, frissítés és mentés. Frissítés és mentés esetén elő fordulhat a korábbi változatok felülírása. Ilyen esetben egy felugró ablakban meg kell erősíteni a szándékot.

A második oszlop elemeinek segítségével az R kód nevű munkalapokat elrejthetjük, ill. ha további szerkesztésre van szükség felfedhetjük és megnyíthatjuk a .R kiterjesztéshez asszociált szerkesztővel (szerencsés esetben ez az Rstudio).

Védelem csoport

Lapvédelem be- ill. kikapcsolása. Ez a két rutin (a Véglegesítés-sel ellentétben) nem módosítja a cellák zárolását.

R ablak csoport

Ez a csoport az R.xls munkafüzetben definiált rutinokat teszi elérhetővé.

Megjelenítés/Elrejtés Ha nincs aktív R példány, akkor a futó R példányok közül választhat egyet a felhasználó. Ha nincs futó R, akkor elindítunk egyet, ha pontosan egy R példány fut, akkor azt fogjuk használni. A gomb megnyomása után a kiválasztott R példány aktív lesz és az ablaka látható.

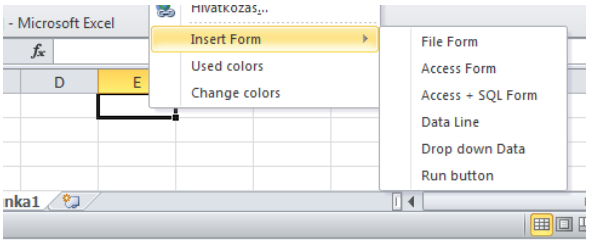
Ha már van aktív R példányunk, akkor a gomb megnyomása után annak ablaka rejtetté válik. Továbbra is használható lesz, de az asztalon nem lesz nyoma.

R-EXCEL összekötés Az aktív R példányban létrehozza a THISXL változót, ami az EXCEL példányra mutató COMObject (Ez tulajdonképpen az EXCEL Visual Basic Application változójának értéke).

Ha nincs aktív R példány, akkor létrehoz egyet, hasonlóan a Megjelenítéshez, de az így létrehozott R példány ablaka alapértelmezésben háttérben marad.

Megszakítás Az R oldalon folyó számolást próbálja megszakítani, az aktív R példány konzoljába küldött ESCAPE karakterrel.

Cella menü



3.2. ábra. A cella menü új elemei.

Insert Form Ez egy menüt kínál fel, amiben a számoló munkafüzetek szokásos elemei találhatóak:

File form Egy három sorból és négy oszlopból álló tartományt tölt ki az aktív munkalapon, melynek bal felső sarka az aktív cella. A tartomány első sora címsor, a második sorban a könyvtár, a harmadikban a file neve adható meg. A második és harmadik sor felhasználói mezőjében található gombok hatására az EXCEL file választó dialógja jelenik meg, így nem kell az elérési utat és a file nevet kézzel beírni. A file típusát egy felugró inputboxban adhatjuk meg. Valójában itt a file filtert kell megadni leírás,*.kiterjesztés alakban. Az EXCEL, .RData, és .R file típusokat a legördülő menüből is ki lehet választani. Ha üresen hagyjuk az inputboxot, akkor tetszőleges file típust választhat majd a felhasználó.

Ha egy már kitöltött File form file típusát, utólag módosítani akarjuk, akkor azt a gombhoz rendelt makróban tehetjük meg.

Access form Hasonló a File form-hoz, azonban egyel több sort tölt ki a munkalapon. Az utolsó sorban az ACCESS adatbázis táblái közül lehet választani.

Access + SQL form Volt olyan korábbi munka, ahol felmerült, hogy nem csak ACCESS adatbázisból, hanem más SQL lekérdezést használó rendszerből is lehessen adatot lekérdezni. Ez az űrlap erre szolgál. A bal felső sarkában van egy gomb, ami két lehetőség között kapcsolgat. Így csak az egyik opció látható. Az ACCESS formhoz képest van egy plusz sor ami az ACCESS lekérdezés where klauzulájának megadására szolgál. Ha átkapcsolunk SQL módra, akkor a kapcsolat létrehozásához szükséges DSN karakterláncot és az SQL parancsot kell megadni. Mivel csak a látható sorokban lévő adatok kerülnek át az R oldalra, az R program el tudja dönteni, milyen adatbázist kell használnia. Pl. ha az SQL parancs R neve lekerdezes1.SQL, akkor a következő kód részlet

```
if(exists("lekerdezes1.SQL")){
    ## kód az SQL ághoz
}else{
    ## kód az ACCESS ághoz
}
```

választ a kétféle adatforrás közül. Mindez csak akkor működik megbízhatóan, ha olyan nevet adunk, ami nem létezik a globális környezetben. Nem célszerű pl. az `x` vagy `i` használata. Egy másik gond az lehet, hogy ha mindkét adatforrást szeretnénk egy-egy futás erejéig kipróbálni. Ilyenkor az `XLdata` függvény `rmHidden` argumentumát is igazra kell állítani, lásd a 7.1 szakaszt. Ezt az `R kód` munkalap `$K$21` mezőjében célszerű megtenni. A képletet célszerű átírni a következő módon:

```
="XLdata(1, 4, rmHidden=TRUE, range='_Rdata')"
```

Ha egy függvényen belül kell ellenőrizni, hogy létezik-e a paraméterként átadott érték, akkor a használhatjuk a `getFromDB` függvényt mintaként:

```
> getFromDB
function (dir, file, table, where = "", dsn, sql)
{
    if (!inherits(try(dsn, silent = TRUE), "try-error")) {
        ODBC.get(dsn = dsn, sql = sql)
    }
    else {
        ACCESS.get(dir = dir, file = file, table = table,
            where = where)
    }
}
<environment: namespace:Rxls>
```

Ekkor, ha az `Adatok` munkalapon a `DSN`, `SQL` sorok el vannak rejtve és az `XLdata`, `rmHidden` paramétere `TRUE`, akkor a függvény az `ACCESS` ágon ellenkező esetben pedig az `SQL` ágon fog végig menni. A módszer az `R try` függvényét használja, ami megpróbálja kiértékelni az első argumentumát (ez jelen esetben a `dsn` argumentum). Ha a kiértékelés hiba nélkül megtörténik, akkor a kiértékelés eredményét kapjuk, ha hiba történt, akkor egy hiba (`try-error` osztályú) objektumot. Az `inherit` függvény ellenőrzi, hogy egy adott érték osztályai között szerepel-e a megadott név. `R`-ben minden változónak van osztálya, vagy explicit (a `class` attribútum értéke) vagy implicit módon (ezek az alaptípusok, pl. `numeric`, stb.)

Az `ACCESS.get` és `ODBC.get` függvények definiálva van az `Rxls` csomagban. Az `ODBC.get` rutint nem tudtam tesztelni az érdekes esetben, amikor `Oracle` rendszerhez akarunk csatlakozni.

`Data Line` A kijelölt tartomány minden sorát kitölti egy-egy adat sorral. Használatával már találkoztunk az 1.3 részben.

`Drop down data` Ha egy adat sorban csak néhány lehetőség közül választhat a felhasználó, akkor célszerű azt egy legördülő menüben felkínálni számára. Erre szolgál ez a menüpont.

Hozzunk létre egy adat sort a `Data Line` beillesztése parancs segítségével, majd a felhasználói beállításnak fenntartott mezőbe írjuk be a lehetőségeket pontosvesszők-

vel elválasztva. Ezután jelöljük ki a cellát és klikkeljünk a jobb egérgombra, majd válasszuk ki a `Drop down data` menüpontot. Ekkor az `R kód` munkalapon létrejön a választék lista az `$A:$A` oszlop végén és a legördülő űrlap alatti cella képlete kitöltődik.

Több cellát is kijelölhetünk. Ilyenkor mindegyik kijelölt cellába egy legördülő menü kerül a cella tartalma alapján. Ha a kijelölt cella üres, akkor a választéklista hosszát kell megadni egy felugró input mezőben. A lista elemeit később az `R kód` munkafüzetben adhatjuk meg.

Az 1.3 részben már láthattunk példát a `Drop down data` alkalmazására.

`Run button` Erre akkor lehet szükség, ha a fő számolás mellett valamilyen rész feladatot szeretnénk külön is végrehajthatóvá tenni. A kijelölt tartomány fölé beilleszt egy gombot `Számolás indítása` szöveggel. A gomb alatti cellák szövegét küldi át az R-nek végrehajtásra. Ez pl. lehet

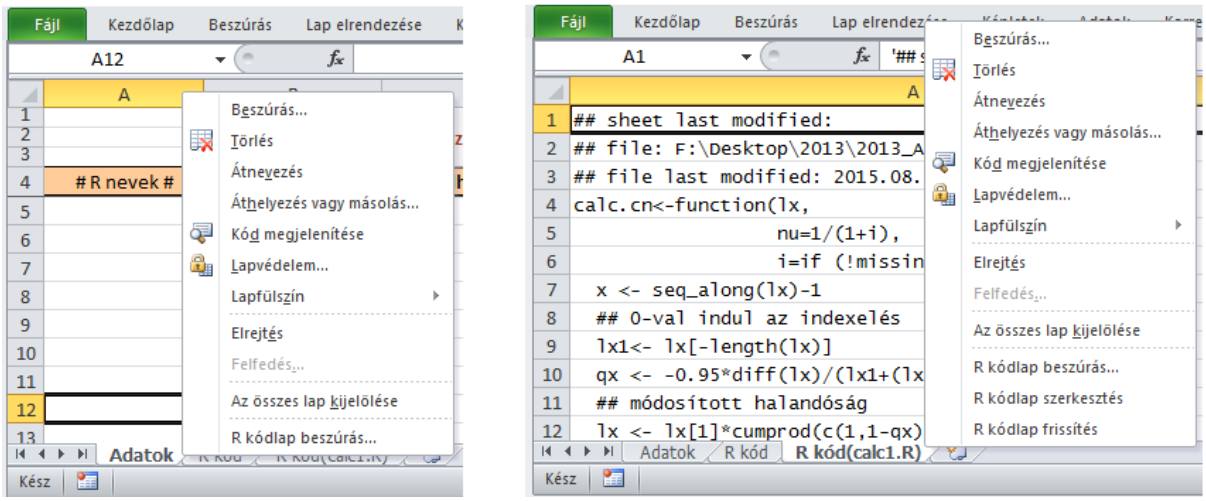
```
="XLsource(<R kódlap neve>, wbname=.wbname)"
```

A `.wbname` R változó értéke aktuális munkafüzet neve a megfelelő formában. A szükséges adatok átadását célszerű az `R kódlapon` elvégezni, pl. az `XLdata` függvény használatával.

`Used colors, Change colors` Kigyűjti, ill. átállítja a munkafüzetben használt színeket. Részletesen lásd a 3.3 szakaszban.

Munkalap fül

Itt is elérhető az `R kódlap` beszúrás parancs számoló munkafüzeteken belül. `R kódlapok` fülén megjelenik még a szerkesztés és frissítés parancs is.



3.3. ábra. A munkalap fül menü új elemei.

3.2. Melyik R példányt használjuk?

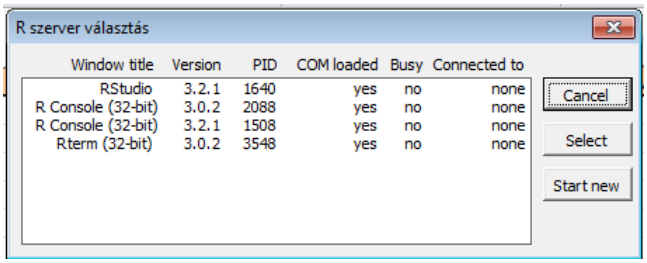
R példány három formában fordulhat elő: terminál ablakban indított `R`, `RGui`, ill. az `Rstudio` `rsession` szála. Mindegyiknek lehet 32–bites ill. 64–bites változata. Az `RGui` két változatban fordulhat elő `MDI` (sok kis ablak egy nagy ablakon belül) ill. `SDI` (sok egymástól független ablak). Az `Rxls` mindegyik változatot tudja kezelni, de ha nincs futó példány, akkor mindig

az RGui-t indítja el, SDI módban. A felhasználói beállítások (esetleges Rprofile file tartalma) nem érvényesülnek, mert a programot -vanilla opcióval indítjuk.

A legegyszerűbb eset, az amikor nincs futó R példány, vagy pontosan egy van. Ha nincs futó R példány, akkor elindítunk egy újat és egyből betöltjük az Rxls, comproxy és com csomagokat.

Ha pontosan egy futó R példány van, akkor azt próbáljuk használni. Ezt lehetne finomítani, mert az általunk indított R példányokban feljegyezzük az EXCEL alkalmazás azonosítóját (hinstance) a .startingapp változóban. Így ha el is veszítettük a kapcsolatot valamilyen oknál fogva le tudjuk ellenőrizni, hogy a futó R példányt mi indítottuk el vagy sem.

Ha több R példány fut éppen, akkor a felhasználónak kell választania egy felugró ablakban. Ekkor is lehetőség van új példányt indítani.



3.4. ábra. R szerver választó dialóg

A választó dialógban megjelenő információk:

windowtitle, PID Az R ablak címsora, ill. az adott R példány process id-je.

COM loaded Azt mutatja, hogy a com csomag be van-e töltve.

busy Ha a com csomag be van töltve, akkor le tudjuk kérdezni a sys.nframe() értéket. Ennek értéke 0, ha az R nem végez számolást, és legalább 1 ha éppen függvény kiértékelés zajlik. Ha a com csomag nincs betöltve a mező értéke ???.

connectedTo Ha a com csomag be van töltve, akkor le tudjuk ellenőrizni, hogy létezik-e a THISXL változó az R oldalon és az a mi EXCEL példányunkra mutat-e? Ez alapján meg tudjuk mondani, hogy az adott R példány a mi EXCEL példányunkkal van összekötve, másikkal, vagy egyikkel sem.

Ha van rá igény a fenti lista esetleg kiegészíthető azzal is, hogy a mi EXCEL példányunk indította-e el az adott R példányt.

3.3. Színek

A számoló munkafüzetek színesek. Számomra segítség, ha látom, hol vannak a munkafüzeten adatok, melyek azok a mezők, amiket ki kell vagy ki lehet tölteni, melyek a feliratok, számolt mezők, stb.. A színeket az Office XP által kínált skálából választottam annak idején, és ezek a színek maradtak a mostani változatban is. Ugyanakkor az újabb Office változatokban már nem csak egy limitált színsorból lehet választani, hanem a teljes szín skála használható.

Ha az Rdev.xlam munkafüzet be van töltve, akkor a cella menü kiegészül két menüponttal, Used colors és Change colors. Mindkét parancs a kijelölt tartománnyal dolgozik. A Used colors a kijelölt tartomány első oszlopát kitölti a munkafüzetben használt színekkel. A Change colors esetében a kijelölésnek két oszlopot kell tartalmaznia. Az első oszlop színeit a második oszlopban mellette található színre cseréli az egész munkafüzetben.

A számoló munkafüzetek R kód munkalapján van egy colortable nevű tartomány. Ennek első oszlopát nem érdemes változtatni. Ha a munkafüzetet átszíneztük, akkor új elem beszúrásakor ebből derül ki, hogy milyen színeket használ a munkafüzet az alapértelmezett színek helyett. A második oszlopot viszont kedvünkre átszínezhetjük és a Change colors menü paranccsal az egész munkafüzetet átszínezhetjük. Itt arra érdemes figyelni, hogy a négy szín különböző maradjon, de legalább a felhasználói adatoknak fenntartott mező színe eltérjen a többitől. Ugyanis a véglegesítés során ez alapján derül ki, hogy mely mezők maradjanak szerkeszthetőek.

Ha egyes részeket manuálisan hozunk létre vagy színezünk át, akkor az utolsó fázisban célszerű egységesíteni a színeket. Ezt a legegyszerűbben úgy tehetjük meg, hogy létrehozunk ideiglenesen egy új munkalapot, arra kigyűjtjük a használt színeket a Used colors paranccsal, a mellette levő oszlopban egységesítünk és a Change colors paranccsal a színeket átállítjuk. Ezután az ideiglenes munkalap törölhető.

Ha valaki az alapértelmezett színeket meg akarja változtatni, akkor azt a Calc.xltm template átszínezésével érheti el.

3.4. Számoló munkafüzetek és az R.xls file

Ha egy számoló munkafüzetet olyan EXCEL példányba töltünk be, amire a Rdev.xlam bővítmény nincs telepítve, akkor a következő történik. Megnyitás után a Workbook_Open szubrutin ellenőrzi, hogy az R.xls-re mutató hivatkozás érvényes-e, ha nem megpróbálja javítani és szükség esetén a felhasználó instrukciókat kap a teendőiről. A menüsor egy új ponttal egészül ki, ez az R menü, melyből az R ablak megjelenítés/elrejtés, R<->EXCEL összekötés és R megszakítás parancsok érhetőek el. Ezek működése azonos a 3.1 szakasz R ablak részében leírttal. A RIBBON felületet használó újabb EXCEL változatokban az R menü a Bővítmények fül alatt található.

4 Az Rxls csomag felhasználói szemmel

Az Rxls csomag exportált függvényei közül valószínűleg azok a legfontosabbak, melyek segítségével adatokat lehet EXCEL-be írni ill. onnan beolvasni. Egy másik csoport a futási állapot kiírásával kapcsolatos. Végül van néhány segédfüggvény is.

4.1. Adat-átviteli függvények

Ezeket a rutinokat már nagyrészt láttuk az 1.3 részben.

XLdata, XLsource Az XLdata függvény segítségével megfelelően formázott EXCEL munkalapról lehet adatokat átvenni. Az XLsource függvény EXCEL-ben tárolt R kódlapot olvas be és értékel ki.

XLreaddf.rows, XLreaddf.cols, XLwritedf Az XLreaddf.cols rutin egy EXCEL táblázatot olvas be R-be. Alapértelmezésben a legfelső sor az R oldalon létrehozott data.frame neveit adja. Az XLreaddf.rows hasonló, de az EXCEL táblázatot először transzponáljuk, azaz az első oszlopban vannak a data.frame nevei, stb. Az XLwritedf függvény egy R data.frame-t ír az EXCEL megadott tartományába. A rutin külön figyel a dátum és factor típusú adatokra. Az NA értékek üres cellaként jelennek meg. Lehetőség van az oszlop szélesség automatikus beállítására (autoFit argumentum) és a kitöltött tartománynak (munkafüzet szintű) nevet adhatunk (setname argumentum).

XLget A megadott EXCEL tartomány tartalmát adja vissza. Közvetlenül valószínűleg ritkán lesz rá szükség.

getFromDB, ACCESS.get, ODBC.get ODBC kapcsolaton keresztül olvas be adatokat. A getFromDB a másik két függvény között választ a paramétereitől függően.

4.2. Futási állapot kiírása

Itt az elképzelés az, hogy az R grafikus eszközéhez hasonlóan van egy logDevice. Amikor az Rxls csomag betöltődik akkor létrejön az 1-es számú ilyen eszköz, ami egy null eszköz. Nevéhez méltóan ez semmit nem jelenít meg. A lehetséges logDevice típusok null, R, EXCEL, TCL.

logDev, logDev.set, logDev.off, logDev.list A logDev függvény új logDevice-t hoz létre a megadott típussal. A logDev.set függvény az aktuális logDevice-t állítja át, logDev.off kikapcsolja azt, míg a logDev.list kilistázza a létező logDevice-okat.

XLwith Ideiglenesen csatolja a keresési úthoz a THISXL(hivatkozás az indító EXCEL példányra), .wbname (munkafüzetnév), .wsname (munkalapnév) vátozókat és a megadott kifejezést

(alapértelmezésben `eval(.FUN)`) végrehajtja (kiértékeli) a munkafüzet saját `R` környezetében oly módon, hogy a `warning` és `stop message` függvények a megadott típusú `logDevice`-ra is írnak.

`background` A megadott kifejezést végrehajtja (kiértékeli) oly módon, hogy a `warning` és `stop message` függvények a megadott típusú `logDevice`-ra is írnak. A különbség az `XLwith` függvényhez képest az, hogy a globális környezetben történik a kiértékelés és ennek megfelelően nem történik meg a fenti változók ideiglenes csatolása a keresési úthoz.

`progress` Egyszerű, szöveges `progreszbárt` hoz létre, ami az aktuális `logDevice`-ra ír.

`logInit` A `logDev` függvény használja. Csak kompatibilitási okok miatt maradt az exportált függvények között.

`logMessage` A `logDevice`-ra ír üzenetet. Valószínűleg sohasem kell közvetlenül meghívni.

4.3. Segéd függvények

`XLDate, XLPOSIXct` Ezt függvényt már láttuk az 1.2 szakasz végén. Ha `XLreaddf` függvényeket használjuk, akkor az eredményben a dátum típusú adatok számként jelennek meg. Ez a függvény a számokat az `R Date` típusú adatává alakítja. Tisztában kell azonban lenni azzal, hogy más dátum típusok is léteznek `R`-ben: `POSIXct` és `POSIXlt`. Ha ilyen típusú adatra van szükségünk, akkor használjuk az `as.POSIXct` és `as.POSIXlt` függvényeket.

További lehetőség, hogy az `XLreaddf` függvényeket `value="value"` argumentummal hívjuk meg. Ezt nem tartom megbízható megoldásnak. Tapasztalatom szerint a nyári időszámítás miatt egy napos eltolódás előfordulhat a dátum típusú adatoknál. Az ok, valahol az időzóna és a nyári időszámítás kezelésében lehet.

```
> x<-seq.Date(as.Date ("2013-01-01"),as.Date ("2014-01-1"),by="1 month")
> df<-data.frame(date=x)
> XLwritedf(df=df,setname="datumok")
NULL
> df["R->EXCEL->R.date"]<-XLreaddf.cols("datumok",value="value")
>
df["eltérés"]<-with(df,as.POSIXct(as.POSIXlt(date),tz="CET")-‘R->EXCEL->R.date‘)
> df["eltérés2"]<-with(df,date-as.Date(‘R->EXCEL->R.date‘,tz=""))
> df[with (df,‘eltérés‘!=0 | ‘eltérés2‘!=0),,drop=FALSE]
         date    R->EXCEL->R.date    eltérés eltérés2
1  2013-01-01 2012-12-31 23:00:00  3600 secs   1 days
4  2013-04-01 2013-04-01 01:00:00 -3600 secs   0 days
11 2013-11-01 2013-10-31 23:00:00  3600 secs   1 days
```

Külön bosszantó, hogy az eredmény még attól is függ, hogyan olvassuk ki az adatokat. Mást kaphatunk, ha csupa dátum típusú adatot tartalmazó tartományt olvasunk be, és mást akkor ha csak néhány cella dátum típusú.

```
> x1<-XLget("datumok")[2:14,1]
> x1<-Rxls:::unlist.keepclass(x1)
> x2<-XLget(XLrange=THISXL$range("datumok")$range("A2:A14"))
> x2<-Rxls:::unlist.keepclass(x2)
> df<-data.frame("név cellával"=x1,
```

```
+                "név cella nélkül"= x2, "eltérés"=x1-x2,
+                check.names=FALSE)
> df[with(df,‘eltérés‘!=0),]
          név cellával    név cella nélkül   eltérés
2  2013-02-01 00:00:00 2013-01-31 23:00:00 3600 secs
3  2013-03-01 00:00:00 2013-02-28 23:00:00 3600 secs
4  2013-04-01 01:00:00 2013-04-01 00:00:00 3600 secs
12 2013-12-01 00:00:00 2013-11-30 23:00:00 3600 secs
13 2014-01-01 00:00:00 2013-12-31 23:00:00 3600 secs
```

Emellett az 1970 előtti dátumok kezelése is problémás a `value` tulajdonság használata esetén. Összefoglalva a `"value2"` tulajdonság használatát javaslom, a dátum értékek utólagos transzformációjával az `XLDate` vagy az `XLPOSIXct` függvényekkel. Utóbbi `POSIXct` formátumba alakítja az adatot, amivel a napon belüli időpont sem veszik el.

`XLScreenUpdateTurnOff` Az `XLwritedf` rutin használja, de egyéb esetekben is szükség lehet rá, ha nagy mennyiségű adatot írunk `EXCEL` munkalapra. Kikapcsolja az `EXCEL` képernyő frissítését és a kalkuláció módját kézire állítja. Ez utóbbi azt jelenti, hogy az adatok írása nem fogja az érintett cellák újraszámolását triggerelni. Visszatérési értéke egy függvény, ami visszaállítja az eredeti állapotot. Tipikus használata:

```
turnOn <- XLScreenUpdateTurnOff(Appl = THISXL)
on.exit(turnOn(), add = TRUE)
```

`XLusedrange` A (`COMObject` formában) megadott tartománynak és a tartomány munkalapján érvényes `usedrange`-nek a metszetét számolja ki. Akkor van jelentősége, ha pl. szeretnénk a `$A:$A` oszlop elemeit beolvasni R-be. A természetesnek tűnő

```
THISXL$range("$A:$A")$value()
```

hívás eredménye egy olyan mátrix, aminek annyi sora van, ahány az `EXCEL` munkalapnak. Ez 32-bites OFFICE esetén is túl sok, 64-bites OFFICE esetén pedig több mint 1 millió sort jelent. Ezt valószínűleg úgy élnénk meg, hogy lefagyott a program, holott csak nagy mennyiségű adatot másol meglehetősen lassan.

`.XLrange` Szöveges címből állít elő hivatkozást (`COMObject`). A cím lehet teljes külső cím, munkafüzettel és munkalappal, vagy a tartomány `EXCEL`ben definiált neve. Ha lehet célszerű a tartományokra névvel hivatkozni.

`attachXL` Csatolja a `THISXL, .wbname, .wsname` változókat a keresési úthoz. A visszatérési érték egy függvény amit a csatolás megszüntetésére lehet használni. Pl. az `XLwith` függvény elese a következő

```
detachXL<-attachXL(env)
on.exit(detachXL(),add=TRUE)
```

Itt `env` a munkafüzet `R` környezete. Ez automatikusan létrejön amikor az `EXCEL` csatlakozik az `R`-hez.

withXLenv Hasonló az R with függvényéhez, bár az argumentumok sorrendje éppen fordított. Kiértékel egy R kifejezést a megadott környezetben. Ha a környezet nincs megadva, de csak egy csatlakoztatott munkafüzet van, akkor annak a környezetét használja a rutin. Tesztelés során lehet hasznos.

detachXLenvs Eltávolítja a keresési útvonalról valamennyi EXCEL környezetet.

win.stop A stop függvényhez hasonlóan leállítja a futást, de a hiba üzenet nem csak az R konzolban, hanem egy felugró ablakban is megjelenik.

pkgzip A megadott és telepített R csomagokat becsomagolja win-binary formátumba.

installRxls Az Rxls csomagban lévő EXCEL fileokat másolja az {APPDATA}/Rxls könyvtárba. Emellett ide kerül még egy versions nevű file is, ami a verzió információt tartalmazza.

getHWND Az R konzol ablakának azonosítóját adja vissza. Az új R.xls munkafüzet nem használja.

install.xls Az R.xls-t használó munkafüzetek telepítője. Az adott csomag EXCEL könyvtárában lévő EXCEL file-okat a kiválasztott könyvtárba másolja és beállítja az R.xls-re történő hivatkozást. Az új számolómunkafüzetekhez nincs rá szükség.

R_RegEntries Az R-rel kapcsolatos regisztry bejegyzéseket jeleníti meg.

4.4. A background és logDev függvény részletesebben

A munkafüzet tipikusan tartalmaz egy gombot, mellyel a felhasználó a számolást elindítja. A gomb makró-hozzárendelése lehet cmdbtnHosszufutas vagy cmdbtnRovidFutas. Ha a gombot a Cella menü -> Insert Form -> Run button paranccsal hoztuk létre, akkor áttételesen de a cmdbtnHosszufutas szubrutint használjuk. Ezek részletes ismertetését lásd az 5.3 szakaszban. Mindkét rutin megkeresi a végrehajtandó kódot. Ez jellemzően a gomb alatti cellából derül ki. A kód egy szöveg. Ezt célszerű munkalap függvényeket használva összerakni valamelyik R kód(<név>) munkalapon. Ezt a szöveget, mint a .FUN szimbólum definícióját átadjuk az R szervernek. Pl. ha az EXCEL-ben kiszámolt kód:

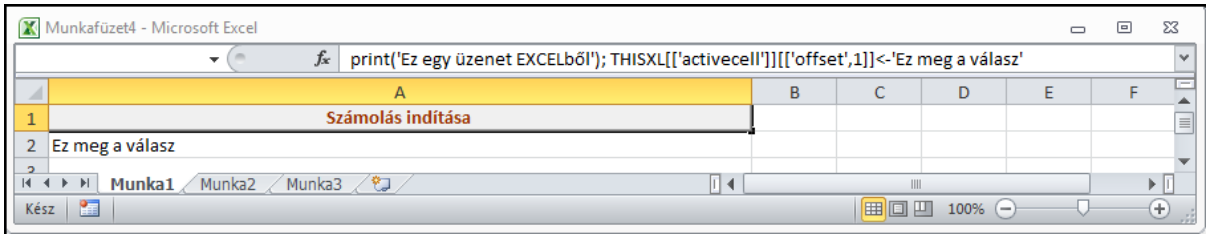
```
print('Ez egy üzenet EXCELből');
withObj(THISXL[['activecell']][['offset',1]],.[['value']]<-'Ez meg a válasz')
```

Akkor a gomb megnyomása után az R oldalon a .FUN szimbólum értéke a következő:

```
> with(data = .XL393694$wb_Munkafüzet4, expr = .FUN)
expression({
    print("Ez egy üzenet EXCELből")
    withObj(THISXL[["activecell"]][["offset", 1]], .[["value"]] <- "Ez meg a
válasz")
})
```

A .FUN változó értékadásához az R oldal COM interfészét használjuk.

A következő lépésben a kódot végre kell hajtani. Attól függően járunk el, hogy a várható futási idő rövid (cmdbtnRovidfutas), vagy hosszú (cmdbtnHosszuFutas). Az első esetben a végrehajtáshoz is használhatjuk a COM interfészt, azonban a második esetben aszinkron hívásra van szükségünk. Ennek az az oka, hogy ellenkező esetben az EXCEL nem fog reagálni a kód futása alatt, ráadásul az EXCEL rendszeres időközönként megjelenít egy figyelmeztető ablakot,



4.1. ábra. A munkalap képe a számolás gomb megnyomása után. A gomb alatti cellában található a végrehajtandó R kód.

miszerint az EXCEL ole művelet befejezésére vár. Ebben az állapotban kilépni sem lehet az EXCEL-ből.

Az aszinkron függvényhívás jelenleg nagyon egyszerűen van megoldva. A kapcsolat inicializálásakor, lekérdezzük az R ablak számát (window handle) és ennek az ablaknak elküldjük a XLwith(env=<XL környezet>)[1] szöveget a Windows üzenet küldési mechanizmusával. Ezt az R úgy érzékeli, mintha a felhasználó beírta volna a konzolba (akkor is, ha egyébként az R ablak el van rejtve) és a kapott kifejezést kiértékeli. Ez a megoldás eddig megbízhatóan működött, de nyilvánvaló, hogy hosszútávon célszerű lenne kiváltani a megfelelő COM mechanizmussal.

Megjegyzendő, hogy az Rstudio használata esetén ez a megoldás nem alkalmazható, mert ez az alkalmazás a kapott üzenetet nem írja be az R konzolba, ezért Rstudio ill. terminál ablakban futtatott R esetén a sendkeys függvényt használjuk.

Az XLwith függvény definíciója a következő:

```
> XLwith
function (expr = eval(.FUN), logdevice = ifelse(exists("THISXL") &&
    comIsValidHandle(THISXL), c("EXCEL", "null", "R", "TCL"),
    c("null", "R", "EXCEL", "TCL")), env = parent.frame())
{
    detachXL <- attachXL(env)
    on.exit(detachXL(), add = TRUE)
    logdevice <- match.arg(logdevice)
    logDev(logdevice)
    on.exit(logDev.off(), add = TRUE)
    withCallingHandlers(eval(substitute(expr), env), error = Rxls:::errMessage,
        warning = Rxls:::condMessage, message = Rxls:::condMessage)
}
<environment: namespace:Rxls>
```

Azaz az expr R kifejezést kiértékeljük, de speciális környezetben. Ha a kifejezés kiértékelése közben, hiba, figyelmeztetés vagy üzenet keletkezik az nem csak az R ablakban, hanem az aktuális logDevice-on is megjelenik. Jelenleg a működés hasonló a rajz felületek (graphical devices) kezeléséhez. A logDev függvény segítségével hozhatunk létre új eszközt, a logDev.off törli az aktív eszközt, a logDev.set pedig átállítja az aktuális eszközt. Jelenleg négy féle eszköz áll rendelkezésre: null, EXCEL, TCL és R.

A null device nevéhez méltóan nem add semmit az R beépített mechanizmusához. Az EXCEL logDevice az EXCEL státuszbárjában jeleníti meg az üzeneteket. A TCL eszköz megjelenít egy ablakot és ezt használja, míg a R eszköz az R konzolját használja.

[1] Ez egy változás a 2.0.250 verzióhoz képest. Ha azt a hibaüzenetet tatpasztaljuk, hogy a rendszer nem találja a background függvényt, akkor frissíteni kell az Rxls csomagot a legújabb változatra. Egyéb tekintetben a korábbi változatokkal készített számoló munkafüzeteknek elvben működnie kell az új Rxls csomaggal.

A logDevice eszköz létrehozásakor az volt a cél, hogy egyszerűen lehessen a futási állapotot megjeleníteni, lehetőleg anélkül, hogy a kódot teletűzdelnénk

```
THISXL[["statusbar"]]<-"...";...;THISXL[["statusbar"]]<-FALSE
```

párokkal.

Megjegyzés. A logDevice elgondolás fokozatosan született meg. A korábban született alkalmazások, vagy közvetlenül használják az EXCEL státuszsorát, vagy a fent leírt megoldás előzetes változatát. A később született alkalmazásokban megjelent a logInit függvény használata. Ez még nem a fent leírt log eszköz. A logInit függvény egy olyan objektumot hozott létre, amivel vagy az EXCEL státusz sorában vagy egy TCL ablakban lehetett szöveges üzenetet ill. progressbar-t megjeleníteni. Jelenleg már az R-ben is elérhető a txtProgressBar ill. a winProgressBar függvény.

4.5. A progress függvény használata

Az a tapasztalatom, hogy egy számolás elindítása után, ha a felhasználó nem kap visszajelzést, hogy valami történik, akkor rövid időn belül elkezdi nyomkodni a billentyűzetet, klikkelget, a jártasabbak pedig a feladatkezelőn keresztül kilövik a programot, hiszen az nem csinál semmit. Ha a számolást EXCEL-ből indítottuk a legkézenfekvőbb az EXCEL státuszbárjában megjeleníteni a számolás aktuális állását. Ezt legegyszerűbben úgy érhetjük el, hogy az előző pontban ismertetett logDev("EXCEL") függvényhívást használjuk. Tesztelés során hasznos lehet a logDev("R"), vagy logDev("TCL") is.

A progress függvény egy új progressbar-t hoz létre az aktuális logDevice-on. Tegyük fel, hogy a számolás különböző fázisai az adatok beolvasása ACCESSből, számolás, majd az eredmények mentése. Ekkor a végrehajtandó (meta) kód a következő képpen alakul.

```
data <- read.access()
res <- do.calc(data)
save.results(res)
```

Ha szeretnénk a felhasználót tájékoztatni az egyes fázisokról, akkor ez kiegészül a következőképpen

```
pb <- progress("%s...", "")
pb$step("ACCESS adatbázis olvasása")
data <- read.access()
pb$step("Számolás")
res <- do.calc(data)
pb$step("Eredmények mentése")
save.results(res)
pb$rm()
rm(pb)
```

Tegyük fel, hogy a számolás hosszú, mondjuk a beolvasott adatok mindegyikén el kell végezni egy műveletet, ami maga is időigényes. Ekkor a do.calc függvény szerkezete a következő lehet

```
do.calc <- function(x) {
    pb <- progress(sprintf(" [%%d/%d]", length(x)), 0, lazyness = 0.3)
    on.exit({
        pb$refresh()
        pb$rm()
    })
    sapply(x, function(y) {
               pb$inc()
               ## egy számolási lépés y-on
           })
}
```

A függvény első sorában létrehozunk egy `progressbar`-t. A `progress` függvény első argumentuma egy minta, ami jelen esetben az `x` vektor hosszától függ. Ha pl. `length (x) = 100`, akkor a minta (`pattern`) értéke `" [%d/100]"` lesz. A mintában szereplő `%d` helyére egy egész érték kerül, aminek kezdőértéke 0, ez a második argumentum.

A `lazyness` paraméter a megjelenítés sűrűségét állítja be. Ha nem adjuk meg, akkor minden változás után frissítjük a `logDevice`-on látható szöveget. Pozitív `lazyness` érték mellett csak akkor frissítjük a `logDevice`-ot, ha az előző frissítés óta eltelt idő (másodpercben mérve) legalább akkora, mint a `lazyness` értéke. A 0.3 másodperces `lazyness` érték mellett a kijelzést folyamatosnak érezzük, de nem írunk feleslegesen sokszor a `logDevice`-ra. A függvényből való kilépéskor frissítjük a `logDevice` szövegét, majd töröljük az eszközről a `pb` `progressbar`-t. Magát a `pb` változót nem kell törölnünk, mivel a függvényből való kilépéskor megszűnik az a környezet, amiben definiáltuk.

Amikor a `pb$inc()` hívást végrehajtjuk, akkor a `progressbar`-ban tárolt értéket eggyel növeljük és megpróbáljuk frissíteni a `logDevice` szövegét. Ez csak akkor fog megtörténni, ha az előző frissítés óta eltelt idő nagyobb mint a `lazyness` érték. Emiatt a kijelző nem folyamatosan számol. Ezért került az `on.exit` rutin elejére egy extra frissítés. Így nem fordulhat elő, hogy a számláló „elakad” 98-nál.

A fenti mintakód esetén a számolás szakaszban a `logDevice` szövege az alábbihoz hasonló:

```
Számolás... [23/100]
```

A `Számolás...` szövegrész a globálisan használt progressbar értéke, míg a `[23/100]` a `do.calc` függvény járuléka.

Gyakran nem a `inc`, hanem a `step` metódust alkalmazzuk egy `progressbarnál`. Ekkor a `logDevice`-on megjelenített szöveg a

```
sprintf(pattern,...)
```

függvényhívás eredménye, ahol a `pattern` az inicializáláskor vagy a `reset` metódus hívásakor megadott minta, `...` pedig a `step` hívás argumentumait jelöli.

5 Az R.xls munkafüzet részletesebben

A legfontosabb részek az Interface, selectfuns és a RIC modulokban vannak. Az Interface modul szubrutinjaiból lehet a számolást indító ill. leállító gomb makróit összerakni. Ezeknek számtalan verziója van, a korábbi számoló munkafüzetekkel való kompatibilitás megőrzése miatt. A selectfuns modulban vannak azok a függvények, amiket a munkalapokon szereplő könyvtár, ill. fileválasztó mezők gombjai használnak. A tényleges munkát a RIC modul végzi. Itt vannak azok a rutinok, melyek az R példányok nyilvántartásáért, elindításáért, leállításáért felelnek.

A további modulok kisegítő függvényekből állnak. A Registry modulban, nevének megfelelően, a registry írás, olvasás rutinjai vannak. Itt van az rutin, ami az R telepítési könyvtárat kiolvassa a registryből, ill. ellenőrzi az RCOM type library bejegyzés meglétét. A WINAPI a windows üzenetküldési mechanizmusát használó rutinok gyűjteménye. A Pick modulban a könyvtár, file, ill. ACCESS tábla választó rutinok vannak. Ezeket a selectfuns modulban használjuk. A worksheetfunctions modul az extAddress munkalapfüggvényt definiálja. Ezt a számoló munkafüzetek használhatják, de érdemesebb munkafüzet szintű neveket használni helyette. Végül a formadjust modult a selectR űrlap összeállításához használtam.

A fenti kód modulok mellett, két class modul is van: Rproc és wndData. Az Rproc osztály példányai egy–egy R folyamatot reprezentálnak a RIC modulban. A wndData típust a WINAPI modul használja.

5.1. A Thisworkbook modul

Ez a modul a következő rutinokból áll.

addRmenu, RemoveRmenu Létrehozza ill. törli a menübár R menüjét. Ennek elemei az R ablak megjelenítés/elrejtés, R <-> EXCEL összekötés, R megszakítás. A R menü a RIBBONt használó újabb változatokban a címsor Bővítmények füle alatt található.

doChecks, checkRcom, checkLang Ezek ellenőrző rutinok. A doChecks a másik két rutint hívja meg. Az Rcom ellenőrzése során megnézzük, hogy RCOM type library regisztrálva van-e. Ha nincs, akkor felugró üzenetben segít abban, hogy a regisztrációt hogyan kell végrehajtani. A felhasználónak lehetősége van a szükséges R kódot vágólapra másolni és azt utána az R ablakba beilleszteni. Közvetlenül EXCEL-ből nem lehet a javítást elvégezni, mert a regisztrációt csak rendszergazdai jogosultsággal lehet elvégezni.

A nyelvi ellenőrzés célja, hogy az ékezetes karakterek a COM mechanizmuson keresztül változatlanul kerüljenek át az R-be és fordítva. Az ellenőrzés nagyon egyszerű. Az R.xls munkafüzet egyetlen munkalapjának $A$1 mezőjében fel vannak sorolva a magyar ABC ékezetes karakterei. Ezt egy VBA változóba másoljuk. Mivel ennek során már a COM mechanizmusa működik, ha nyelvi beállítások nem megfelelőek, akkor az őű betűk helyett ou-t kapunk. Hiba esetén a felugró ablak tájékoztat arról, mit kell átállítani.

Előfordulhat, hogy a vezérlőpult felépítése némileg változik. Ekkor a felugró ablak szövegét, ami a modul elején lévő szöveges konstans célszerű átírni. A lényeg, hogy az unicode szabványt nem támogató programok nyelvét kell magyarra állítani. Ez első sorban angol (vagy egyéb nem magyar) nyelvű Windows rendszereknél lehet gond.

Workbook_BeforeClose, Workbook_AddinUninstall Bezáráskor az R menüt eltávolítjuk és az adott EXCEL példány által elindított R példányokat megpróbáljuk leállítani a RIC modul CloseRcon eljárásával. Egészen pontosan a következő történik. Végigmegyünk a futó R példányokon.

Ha valamelyik R példány ablaka rejtett, és a com csomag nincs betöltve, akkor láthatóvá tesszük az ablakot. Így a felhasználónak lehetősége lesz kilépni az R alkalmazásból és bezárni az ablakot. Ha az adott R példányt a mi EXCEL példányunk indította el, akkor megnézzük számol-e még. Ha igen akkor a felhasználó dönthet, hogy megpróbálja-e leállítani az éppen folyó számolást.

Igen válasz esetén, megszakítjuk a számolást és leállítjuk a R példányt. Nem válasz esetén csak a kapcsolatot reprezentáló környezetet töröljük és kilépünk az EXCELből. Ez a legproblémásabb eset. Előfordulhat, hogy az éppen futó R kódban más hivatkozás is van az EXCEL példányunkra, vagy annak valamely részére. Ilyenkor az EXCEL-ből látszólag kilépünk, de az a háttérben tovább fut.

A felhasználó választhatja a Mégse gombot is. Ekkor az EXCEL leállítása megszakad.

Abban az esetben, ha egy R példányt nem a mi EXCEL példányunk indította el, de van a mi EXCEL példányunkra mutató környezet, akkor csak ezt töröljük.

Workbook_Open Elvégzi az RCOM és a nyelvi beállítás ellenőrzését és beállítja az R menüt.

Rmenuset Attól függően, hogy az aktuális R példány ablaka látható vagy rejtett (esetleg nem létezik) vált az R menü `R ablak elrejtése` ill. `R ablak megjelenítése` lehetőségei között.

isXP Ez a függvény igaz értéket ad vissza, ha az operációs rendszer Windows XP. Jelenleg nem használjuk.

5.2. A Pick modul

PickACCESSTable(defaultDir, file) A megadott ACCESS adatbázis tábláinak listájából választhat a felhasználó. Ehhez elindítunk egy ACCESS példányt, betöltjük az adatbázist, majd selectMDBtable űrlapot kitöltjük, ebből választhat a felhasználó.

PickFile(xfilter As String, defaultDir As String, Optional title) File választás. Lehetőség szerint az OFFICE beépített file választó dialógját használjuk. Az xfilter sztring "leírás_1,minta_1,leírás_2,minta_2,...,leírás_n,minta_n" alakú, ahol a minta *.<ext> valamilyen kiterjesztéssel. Ez lehet pl. *.xl* is.

GetDirectory(initDir, Msg) Ez a könyvtár választó rutin. Az initDir a kezdő könytár, míg az Msg az felugró ablak címsora. Lehetőség szerint az OFFICE szokásos file választó dialógját használjuk, korábbi változatok esetén pedig a windows shell.application browseforfolder dialógját (a GetDirectorySH rutinban). Az utóbbi eset a mai rendszereken gyakorlatilag nem fordul elő; ha mégis, akkor a kezdő könyvtár argumentumot nem használjuk.

5.3. Az Interface modul

Ebben a modulban vannak az R kód futtatására szolgáló rutinok. Jellemzően ezek közül kerül ki a futtatást indító gomb makróhozzárendelése.

RunRmain() Az új számoló munkafüzetek ezt használják a számolás indítása gomb makrójaként. Az R kód munkalapon van az Rmain nevű tartomány. A szubrutin az itt található kódot küldi át az aktív R példánynak kiértékelésre. A kiértékelés aszinkron módon történik, azaz az EXCEL nem vár a számolás befejeződésére.

RIgnore() Az aktív R példány számolásának megszakítása. Ehhez a RIC modul stopR rutinját használjuk. Ha az adott R példány számolást végez, akkor megjelenik egy figyelmeztető ablak. Ha a felhasználó megerősíti, hogy meg akarja szakítani a számolást, akkor ESCAPE karaktert küldünk az R ablakba. Ennek hatására a számolás megszakad, gyakran némi késleltetéssel.

PushCmdToR(cmd As String) Az R oldal .FUN változóját a cmd-ben megadott értékre állítja és a THISXL változót frissíti.

cmdbtnRovidFutas(Optional cmd As String = "") Hatására az R a cmd kifejezést szinkron módon kiértékeli, azaz az EXCEL mindaddig nem reagál, amíg az R be nem fejezte a kiértékelést.

cmdbtnHosszuFutas Az előző aszinkron változata. Paraméterek: cmd, logdev, initmsg.

RovidFutas(Optional caption As String = "*") A caption feliratú gombot megkeresi az aktív munkalapon, majd a gomb alatti területben levő szöveget a cmdbtnRovidFutas szubrutin segítségével átadja az R-nek kiértékelésre. A korábban készült munkafüzetekben ezt használtam.

HosszuFutas(Optional caption As String = "*") Az előző aszinkron változata.

getButton(caption = "*", macro = "*", sheet = Nothing) As Shape Mindegyik argumentum opcionális. A sheet munkalapon (ha nincs megadva, akkor az aktív lapon) megkeresi azt gombot (shape), melynek makróhozzárendelése és felirata illeszkedik az argumentumok között megadott mintára. Az összehasonlítás a like operátorral történik. Az újabb munkafüzetekben nem használjuk.

GetCmd(caption = "*", brc As range, tlc As range) Mindegyik argumentum opcionális. brc ill. tlc a bottomrightcell ill. topleftcell rövidítése. Ha ezek nincsenek megadva, akkor a caption minta alapján keresünk egy gombot az aktív munkalapon és ennek bal felső, ill jobb alsó sarka alatti cellára állítjuk a brc, tlc változókat. Ezután a tlc:brc tartományban megkeressük az első nem üres cellát ennek értéke lesz a függvény visszatérési értéke. Ha a tlc egy egyesített tartomány eleme, akkor az egyesített tartomány (azaz a jobb felső sarkában lévő cellájának) értékét adja vissza a függvény.

5.4. A RIC modul

A RIC (R internal connector) a legfontosabb modul az R.xls munkafüzetben. Ez teszi lehetővé, hogy feladatokat adjunk valamelyik futó R példánynak. A modul private, azaz rutinjai csak az R.xls munkafüzet számára láthatóak. A modul legfontosabb függvénye az Revalsync. Ez elküldi a megadott utasítást az aktív R példánynak. Ha nincs rendelkezésre álló R szerver, akkor hiba generálódik, és a hiba kezelő rutin megpróbál egy alkalmas R példányt találni, majd újra próbálkozik az utasítás elküldésével.

A modul három globális változót használ:

allRproc Ez egy Rproc collection. A változó inicializálásakor a rendszerben lévő valamennyi R példány adatait összegyűjtjük és itt tároljuk.

curRproc Az általunk használt R példány.

Rmayfail A hiba kezelő rutin használja. Ha értéke IGAZ, akkor nem erőltetjük a feladat végrehajtását, ha hamis, akkor hiba kezelő rutin mindent megtesz, hogy találjon egy alkalmas R példányt. Az igaz értéket pl. akkor használjuk, ha az EXCELből való kilépés előtt szeretnénk törölni a THISXL változót az R oldalon. Ha valamilyen oknál fogva az R már nem fut, akkor nem kell elindítani egy új példányt, csak azért, hogy töröljük az éppen létrehozott THISXL változót.

A modul rutinjai:

getAllRproc A Windows winmgmts:\\.\root\cimv2 objektumát használva lekérdezzük valamennyi futó R példányt. Minden R példányt egy Rproc objektum reprezentál. Ebben számos tulajdonságot feljegyzünk:

```
Public Name As String             'A program neve: RGui, rsession, Rterm
Public RIC As InternalConnector   'Az R COMservere, ha van
Public hwnd As Long               'Az R konzol ablaka
Public outerHwnd As Long          'Az R-et futtató program ablaka
Public wndColl As Collection      'Az R példányhoz tartozó összes ablak
Public title As String            'Az ablak címsora
Public pid As Long                'Az R processid-je
Public reportedPID As Long        'Az R Sys.getpid() hívásának értéke
Public useSendkey As Boolean      'Sendkeys függvényt kell-e használni
Public sendkeyPrefix As String    'Előtag a sendkeys függvényhez
Public delay As Long              'Késleltetés a sendkeys használatakor
Public useClipboard As Boolean    'Clipboardot kell-e használni
Public caption As String          'A cím és a PID kombinációja
```

A fenti elemek közül a useSendkey, sendkeyPrefix, delay és useClipboard értékeket az R process neve alapján döntjük el. Végül a setCOMServers eljárással az RIC értékeket gyűjtjük össze.

setCOMServers Itt azt használjuk ki, hogy az R COM szerverét le és fel tudjuk kapcsolni a comRegisterServer ill. comUnregisterServer függvényekkel. Az eljárásunk a következő. Kérünk egy rcom szervert a rendszertől, ha kapunk egyet akkor lekérdezzük a PID-t és ez alapján az rcom szervert az allRproc collection megfelelő elemében feljegyezzük. Ezután a szervert lekapcsoljuk és kérünk egy következőt, mindaddig, amíg az összes rcom szervert össze nem gyűjtöttük és le nem kapcsoltuk. Ezután a szervereket visszakapcsoljuk.

RConnectedToThis() As VbTriState Lekérdezzük az R THISXL változóját és összehasonlítjuk az EXCEL Application változójával.

RnotBusy Variant értékű függvény. A COM interfészt használja. Ha az nem áll rendelkezésre, akkor értéke empty, különben IGAZ vagy HAMIS.

Rprocdata Az R szerver választó dialóghoz gyűjti össze az adatokat.

RprocInit Először összegyűjti az összes R proceszt a getallRproc függvénnyel. Ha csak egy R példányt találunk, akkor azt állítjuk be aktív R példánynak (curRproc). Ha többet is találtunk, akkor a felhasználó választhat.

Rstart Ezzel a rutinnal indítunk el egy új R példányt. A regisztryből kérdezzük le az R telepítési útvonalát, és az RGui programot -sdi és -vanilla opciókkal indítjuk el. A R_DEFAULT_PACKAGES környezeti változó értékét átállítjuk a következőre

```
datasets,utils,grDevices,graphics,stats,methods,Rxls
```

Ez a szokásos esetben betöltött csomagok listája kiegészítve a Rxls csomaggal. Így elindulás után az R COM szervere rögtön rendelkezésünkre áll.

Indítás után 20 kísérletet teszünk arra, hogy az allRproc collectiont feltöltsük. Minden kísérlet között 1/10 másodpercet várunk. Ez a beállítás az általam használt gépeken megbízhatóan működött. Nagyon lassú, régi konfigurációk esetén meg lehet növelni a várakozást, vagy az ismétlések számát.

Ha az R példányt megtaláltuk, akkor a COM szerverét próbáljuk megkapni. Ez csak akkor áll rendelkezésre, ha com csomag betöltése megtörtént. Ez némi időbe telik az R elindulása után, ezért itt megint csak 20 kísérletet végzünk a setCOMServers eljárással 1/10 másodperces késleltetésekkel. Ha sikerül a COM szerverhez csatlakozni, akkor az R oldalon beállítjuk a .startingApp változót a hinstance értékre. Ez egy karakterlánc, amit az EXCEL példány Hinstance tulajdonságából nyerünk a hinstance függvénnyel.

loadCOM Ezt a rutint akkor használjuk, ha olyan R példányhoz akarunk csatlakozni, aminek COM szervere nem áll rendelkezésre. Az eljárásunk hasonló a Rstart szubrutin második felében követetthez. Először az R példány ablakába írjuk a

```
require(Rxls);comRegisterServer()
```

szöveget. Ezzel betöltjük a szükséges csomagokat (comproxy, com, Rxls), ha azok nem voltak betöltve, és felkapcsoljuk az COM szervert, ha az le volt kapcsolva. Ezután a 20 kísérletet teszünk 1/10-ed másodperces idő közökkel a COM szerverhez való csatlakozásra. Ebben a lépésben a setCOMServers szubrutint használjuk.

Rclose Az R oldalon töröljük az EXCEL példányra való hivatkozást. Itt az aszinkron interfészt használjuk. Megfontolandó, hogy kiváltható-e ez a COM szerver használatával.

Rgetsymbol, Rsetsymbol, Reval, REvalSync, correct Ezek a függvények az aktív R példány COM szerverét használják a nevüknek megfelelően. A különbség az Reval és az RevalSync között az, hogy első a kiértékelés eredményét visszaadó függvény, míg a második a kiértékelés eredményét eldobó szubrutin.

Ezeknek a rutinoknak a correct hibakezelő rutin a lelke. Ha COM szerveren végrehajtott művelet, valamilyen okból meghiúsul és hibát generál, akkor a correct rutin próbálja helyrehozni a hibát. Ezután a műveletet megismételjük.

A következő hibákat kezeli a rutin:

Az aktív R példány, vagy annak COM szervere hiányzik Ha az allRproc collection nincs inicializálva, akkor a RprocInit rutinnal feltöltjük azt, ha több futó R példány is van, akkor a felhasználó választhat aktív R példányt, stb.

Ha nincs futó R példány, akkor egy nulla hosszúságú collection keletkezik ebben a lépésben, és az ismételt próbálkozásnál ugyanerre az ágra kerülünk, de most már inicializált nulla hosszúságú allRproc collectionnal. Ekkor elindítunk egy új R példányt az Rstart eljárással és újra próbálkozunk a művelet végrehajtásával.

Ha van ugyan aktív R szerverünk, de annak ablaka már nem élő, akkor erről tájékoztatjuk a felhasználót, igény esetén inicializáljuk a allRproc collectiont és újra próbálkozunk vagy ha erre nincs igény, akkor leállunk.

Végül, ha az aktív R ablak élő, de mégsem tudunk csatlakozni a COM szerveréhez, akkor megpróbáljuk betölteni és felkapcsolni a COM szerverét a loadCOM eljárással, majd újra próbálkozunk a művelet végrehajtásával.

Az aktív R szerver COM szervere nem elérhető Ekkor a setCOMServers rutinnal lekérdezzük az elérhető COM szervereket és újra próbálkozunk a művelet végrehajtásával. Ez tipikusan akkor fordul elő, ha az R-ből kiléptünk. Ilyenkor az EXCEL oldalon olyan szerverre hivatkozunk, ami már nem létezik. Az ismételt próbálkozás szintén hibát fog generálni, de most már az első ágon fogjuk azt kezelni. Mivel az aktív R példány már nem létezik, annak ablaka sem élő. Ez azt eredményezi, hogy a felhasználó kap egy kérdést, hogy akar-e új R szervert választani, stb.

Egyéb hiba A hiba szövegét egy felugró ablakban ismertetjük a felhasználóval és leállunk.

REvalAsync, cmdToR Az REvalAsync jelenleg csak egy másik név a cmdToR függvényre, ami az R konzolba írja az argumentumként kapott szöveget. Háromféle eljárást használunk az R változattól függően.

RGui Ekkor a WINAPI modul sendtext eljárását használjuk, ami közvetlenül a Windows üzenetküldő mechanizmusára épül.

RStudio Ekkor az utasítás szövegét a clipboardra másoljuk, és a Windows Sendkeys lehetőségét használva betűparancsok segítségével beillesztjük a clipboard tartalmát a konzol ablakba.

Rterm Ekkor csak a Sendkeys lehetőséget használjuk.

connect Beállítja az R oldal THISXL változójának az értékét.

CloseRCon Ezt a rutint az EXCEL leállítása során hívjuk meg. Ekkor végignézzük a futó R példányokat.

Ha valamelyik R példány ablaka rejtett, és a com csomag nincs betöltve, akkor láthatóvá tesszük az ablakot. Így a felhasználónak lehetősége lesz kilépni az R alkalmazásból és bezárni az ablakot.

Ha az adott R példányt a mi EXCEL példányunk indította el, akkor megnézzük számol-e még. Ha igen akkor a felhasználó dönthet, hogy megpróbálja-e leállítani az éppen folyó számolást.

Igen válasz esetén, megszakítjuk a számolást és leállítjuk a R példányt.

Nem válasz esetén csak a kapcsolatot reprezentáló THISXL változót töröljük és kilépünk az EXCELből.

Ez a legproblémásabb eset. Előfordulhat, hogy az éppen futó R kódban más hivatkozás is van az EXCEL példányunkra, vagy annak valamely részére. Ilyenkor az EXCEL-ből látszólag kilépünk, de az a háttérben tovább fut.

A felhasználó választhatja a Mégse gombot is. Ekkor az EXCEL leállítása megszakad.

Abban az esetben, ha egy R példányt nem a mi EXCEL példányunk indította el, de erre az EXCEL példányra mutat a THISXL változó, akkor csak a THISXL változót töröljük.

interruptR Ha van aktív R példány, akkor az ESCAPE karaktert írjuk az ablakába.

StopR Az R menü R megszakítás rutinja használja. Ha az R dolgozik, akkor rákérdezünk, hogy megszakítsuk-e a műveletet, és igen válasz esetén a interruptR függvénnyel ezt megkíséreljük.

RBringForeground Az aktív R példányt előtérbe hozza. Ha nincs aktív R példány, akkor létrehoz egyet.

RPutBackground Ha előtérben van az aktív R példány, akkor elrejti az ablakát.

ChangeRhwndState Ha látható az aktív R példány akkor elrejti, ha nem látható akkor előtérbe hozza a Rbringforeground eljárással.

BringAllRForeground Az allRproc kollekcióban szereplő R példányokat mind előtérbe hozza.

curRVisible Igaz értéket ad vissza, ha az aktív R példány ablaka nincs elrejtve.

hinstance Egyedi azonosítót ad vissza. Azért van külön függvényben, mert attól függően, hogy régi vagy új EXCELünk van más kódra van szükség.

screenUpdate Visszaállítja az EXCEL ScreenUpdating tulajdonságát igazra.

quote Ha a Sendkeys hívást használjuk, akkor a szöveg speciális karaktereit kapcsos zárójelek közé kell tenni. Ezt az átkódolást végzi a rutin.

dummyset Az Rgetsymbol, Reval függvények használják. Csak azért van rá szükség, mert értékadáskor más kód kell objektum és nem objektum érték esetén.

5.5. A WINAPI modul

Itt vannak a Windows API deklarációk és az azokra épülő rutinok. A korábbi változatokban ez a rész hangsúlyosabb volt, mert a futó R példányok feltérképezése az enumwindows API híváson keresztül történt. Ez a rész a modul végén kikommentezve megtalálható.

sendkeysToHwnd, sendText Az első a Sendkeys függvényt, a második a Sendmessage hívást használva küld szöveget a megadott ablakba.

GetWndPid Ez egy wrapper a GetWindowThreadProcessId API hívás körül. Adott ablak pidjét adja vissza.

getAllWndFromPid, enumAllbyPid Az enumWindows API hívást használva, lekérdezzük az adott processhez tartozó valamennyi ablakot.

GetWndText Adott ablak címsorának szövegét adja vissza.

EnumWndLike, childWndLike Adott ablak child ablakai között megkeresi az adott mintára illeszkedő címűt. Az összehasonlítást a like operátorral végezzük. Az childwindow-kat az enumChildWindows hívással járjuk végig az első találatig.

Az itt szereplő rutinokat a RIC modulból ill. az Rproc objektumok inicializálásakor használjuk. Néhány API függvényt, közvetlenül meghívunk a RIC modulból, pl. sleep, isWindow, isWindowVisible, showWindow.

5.6. A selectfuns modul

Az Rdev.xlam bővítmény segítségével File, ACCESS, vagy ACCESS + SQL választó elemeket illeszthetünk be a munkafüzetekbe. Ezeknél a felhasználói mező jobb felső sarkában megjelenő kis gombokhoz rendelt makrók az ebben a modulban levő rutinokat használják. Ezeket a makrókat az Rdev.xlam bővítmény írja a számoló munkafüzetbe. Pl. ha File választó mezőt szúrunk be, akkor a munkalap makrói között az alábbihoz hasonlóak jelennek meg.

```
Private Sub File2Click()
  selectFile Me.Shapes("Button 2").TopLeftCell, "R,*.R"
End Sub
Private Sub mappa1Click()
  selectmappa Me.Shapes("Button 1").TopLeftCell
End Sub
```

A `select...` szubrutinok a `R.xls` munkafüzetben vannak definiálva. Jelenleg a következőket használjuk:

`selectMappa(cell As range)` Könyvtár választó dialógot jelenít meg. Ha a felhasználó választ egy könyvtárat, azt a `cell` cellába írjuk. A kezdő könyvtárat a `cell` cellától jobbra lévő cellából vesszük.

`selectFile(cell As range, filter As String)` File választó dialógot jelenít meg. A kezdő könyvtár értékét a `cell`-től jobbra, fent lévő cellából veszi. Ha a felhasználó a `Mégse` gombot nyomja meg, akkor nem változtat a `cell` cella tartalmán. Ha a felhasználó választott egy file-t, akkor annak neve, az könyvtár elérési útja nélkül a `cell` cellába kerül, míg a `file`-t tartalmazó könyvtár elérési útja a `cell` cella fölötti cellába íródik.

`selectMDB(cell As range)` A `selectFile` rutint hívja meg `"Access,*.mdb"` filter értékkel.

`selectTBL(cell As range)` Access tábla választó dialógot jelenít meg. Ehhez a `Pick` modul `PickACCESSTable` függvényét használjuk. Az `Access` adatbázis elérési útját a cell cellától jobbra fent lévő két cella értékéből rakja össze.

`selectSQL(cell As range, bl1, bl2)` Ez a szubrutin az `ACCESS + SQL` választó mezőkben használatos. A `cell` cella alatti `bl1` számú sort elrejti ha az látható és fordítva, majd az ez alatti `bl2` számú sort elrejti ha az első `bl1` sor látható és fordítva.

5.7. Egyebek

`formadjust` modul

A `selectR userform` beállítására szolgál. Valójában nincs rá szükség az `R.xls` munkafüzetben, viszont így volt a legegyszerűbb a `selectR` űrlap elemeit egymáshoz igazítani.

`Registry` modul

A modul nagy részét valahonnan másoltam, már nem emlékszem honnan. Az alapvető registry műveleteket teszi lehetővé: olvasás, írás, törlés. Mi csak olvasásra használjuk. Erre épül a modul `RcomRegistered` függvénye, amit az `R.xls` munkafüzet betöltésekor a telepítés ellenőrzésére használunk, ill. `RFromReg` függvény, ami az `Rgui` program elérési útját számolja ki.

`Worksheetfunctions` modul

Egyetlen, munkalapokon is használható függvényt definiál. Az `extAddress`-szel már találkoztunk az 1.3 részben. Ennek a függvénynek két argumentuma van, egy tartomány és `quote` nevű logikai változó. A tartomány teljes nevét (`munkafüzet!munkalap+cella cím`) adja vissza. Ha a `quote=IGAZ`, akkor aposztrófok között, különben anélkül. Ahhoz, hogy mindig a megfelelő értéket kapjuk az `R.xls` munkafüzet az `EXCEL` alkalmazás `WorkbookBeforeSave` eseménykezelőjét kiegészíti. Az történik, hogy végigmegyünk a mentendő munkafüzet lapjain és azokat a cellákat, melyek formulája hasonlít a `*extaddress*` mintára (a `like` operátort használva)

dirtyként megjelöljük. Ezután a munkalapot kiértékeljük a calculate metódussal. Munkafüzet mentése közben a státuszsorban felvillanó szöveg jelzi, hogy a eztAddress függvények ellenőrzése történik.

A megvalósítás úgy történik, hogy a R.xls munkafüzetben egy APP nevű Class modult is létrehozunk. Ebben van egy WithEvent kulcsszóval definiált application objektum, amit inicializáláskor Excel.Application értékre állítunk. Az iniciálizálás az R.xls munkafüzet megnyitásakor történik meg. Ez a megoldás csak azt az esetet kezeli, ha a munkafüzet neve, az EXCEL mentés másként funkciója nyomán változik meg. Ha a munkafüzet neve valamely egyéb módon változott meg, akkor a megnyitás utáni mentés helyreállítja az extAddress-re épülő hivatkozásokat.

Egy másik lehetőség a APP modulban definiálni a WorkbookOpen eseménykezelőt is, a fentihez hasonló módon.

6 Az `Rdev.xlam` bővítmény részletesebben

Ebbe a munkafüzetbe gyűjtöttem azokat a rutinokat, melyek csak a fejlesztéshez szükségesek, lásd a 3.1 szakaszt. Néhány olyan rutin is ide került, ami a dokumentáció elkészítésében segített. Ez a fejezet a munkafüzet felépítését ismerteti.

6.1. A `finalize` modul

Itt vannak a számoló munkafüzet véglegesítésénél használt rutinok.

`FinalizeWb` Először feloldjuk a lapvédelmet (`unprotectSheets`), majd elrejtjük az `R` kódot (`HideRcodes`), a felhasználói cellák zárolását feloldjuk (`lockcells`), a bekapcsoljuk a lapvédelmet (`protectSheets`), végül elmentjük a munkafüzetet.

`HideRcodes, unHideRcodes` Elrejtésnél végigmegyünk a munkalapokon. Ha az `isToHide` függvény azt mondja egy munkalapról, hogy el kell rejteni, akkor elrejtjük és a `DEBUG/UNDEBUG` választéklista értékét `UNDEBUG`-ra állítjuk. Ellenkező esetben megnézzük, hogy az adott munkalap nevei között szerepel-e az `R_names`. Ha igen, akkor ennek az oszlopát is elrejtjük. Ez utóbbit az `Adatok` munkalapon használjuk. Visszaállításnál, csak az elrejtett munkalapokat és oszlopokat tesszük láthatóvá, nem változtatunk a `DEBUG/UNDEBUG` mező értékén.

`protectSheets` Az aktív munkafüzet látható munkalapjainak védelmét bekapcsoljuk.

`unprotectSheets` Az aktív munkafüzet valamennyi munkalapjának a védelmét kikapcsoljuk.

`lockcells` Az aktív munkafüzet látható munkalapjain a használt tartomány celláinak `locked` tulajdonságát állítja be a cella színe alapján. Ha a cella színe (pontosabban `colorindexe`) azonos a `getCustomColor` értékkel, akkor a felhasználó írhat a cellába, ellenkező esetben nem. A rutin végén van egy rész, ami `checkbox` vezérlők használatát teszi lehetővé. Ha a vezérlő `linkedcell` cellája ugyanazon a munkalapon van, akkor ez a cella sem lesz zárolt. Ez a rész valószínűleg fölösleges.

`isDebugDD(sh As Worksheet, dd As DropDown) As Boolean` A `dd` `DropDown`-ról ellenőrzi, hogy a `DEBUG/UNDEBUG` választóról van-e szó.

`isToHide(name As String) As Boolean` A `name` paraméterről ellenőrzi, hogy `R kód*` vagy `*hidden*` alakú-e.

`getCustomColor` Kikeresi a felhasználói mezők színét, pontosabban a `colorindex` tulajdonságát az `R kód` munkalap `colortable` tartományából.

6.2. Az insertform modul

insert... rutinok

insertAccessFrom, insertAccessSQLForm, insertFileForm, insertDataLine Ezek a rutinok egy kaptafára mennek. A Rdev.xlam munkafüzet samples munkalapján lévő mintákat átmásoljuk a számoló munkafüzetbe, ha szükséges kis méretű gombokat illesztünk be a megfelelő cellákba és ha kell a R.xls munkafüzetre mutató hivatkozást beállítjuk.

Az insertDataLine rutin kivételével egy elemet illesztünk be egyszerre az activeCell cellába (ez a kijelölés bal felső sarka). Az insertDataLine a kijelölés minden sorába beilleszt egy Data Line elemet.

insertDropDown A kijelölt cellákba (pontosabban a kijelölés első oszlopának celláiba) beszúr egy-egy Drop Down űrlapot. A választék listát pontos vesszővel elválasztott szövegként a cella értékeként adhatjuk meg. Ha ez hiányzik, akkor egy felugró ablakban a lista hosszát kell megadni. Ez esetben utólag ki kell tölteni az R kód munkalap megfelelő tartományát. A tartomány fölé az Adat neve van beképletezve, ez a cellától balra álló mező értéke, mellé pedig a választott érték (Ide mutat az űrlap linkedcell tulajdonsága).

A rutin gondoskodik arról, hogy a Drop Down űrlap alatti cella képlete a választék lista aktuális értékét tartalmazza és színe a tőle jobbra álló cella színével egyezzen meg. Így véglegesítés után a felhasználó nem tudja elrontatni a képletezést.

insertCalcBtn (optional extra) Beszúr egy számolást indító gombot a kijelölt tartomány fölé. A gomb szövege Számolás indítása és makró hozzárendelését a newRun függvény számítja ki a gomb azonosítójából. Az eredmény alábbihoz hasonló

```
Private Sub Btn1run()
  runBtnClick Me.Shapes("Button 1")
End Sub
```

A runBtnClick az R.xls munkafüzet Interface moduljában van definiálva és a shape alatti tartomány szövegét küldi el kiértékelésre az R-nek.

Ha a insertCalcBtn szubrutint nem üres extra paraméterrel hívjuk meg, akkor az így megadott szöveg a runBtnClick sor végére kerül. Ha például extra=",True", akkor a gomb makró-hozzárendelése

```
Private Sub Btn1run()
  runBtnClick Me.Shapes("Button 1"), True
End Sub
```

alakúra változik. Az első változat aszinkron módon hajtja végre a kódot, míg a második szinkron módon. Az extra paramétert a dokumentáció készítése közben használtam.

Segéd rutinok

formcopy(cell As Range) Ezt az eljárást a insert...form nevű rutinokból hívjuk, miután az Rdev.xlam munkafüzet samples munkalapján a megfelelő részt másolásra kijelöltük. A cell tartományba illesztjük a másolandó rész formuláit és formázását. A végén a beillesztett részt kijelöljük a select metódussal és a színeket a setcolors rutinnal a számoló munkafüzettel összhangba hozzuk.

insertSelectBtn(vbcs as Collection, c As Range, which As String, Optional exarg = "")
A c cella bal felső sarkába illeszt egy kis méretű gombot a smallBtnAt eljárás segítségével. A gombhoz a which paraméter alapján megfelelő makrót rendel. A newSelect szubrutinnal kiszámolt makrót feljegyezzük a vbcs collectionban. Az egyes típusok esetén a következő szubrutinok lehetségesek

```
Private Sub File7Click()
  selectFile Me.Shapes("Button 7").TopLeftCell, "R,*.R"
End Sub
Private Sub mappa6Click()
  selectmappa Me.Shapes("Button 6").TopLeftCell
End Sub
Private Sub SQL5Click()
  selectSQL Me.Shapes("Button 5").TopLeftCell, 4, 2
End Sub
Private Sub TBL4Click()
  selectTBL Me.Shapes("Button 4").TopLeftCell
End Sub
Private Sub MDB3Click()
  selectMDB Me.Shapes("Button 3").TopLeftCell
End Sub
```

A select... szubrutinok a R.xls munkafüzet selectfuns moduljában vannak definiálva.

smallBtnAt (c as range) A c tartomány bal felső sarkába elhelyez egy kis méretű gombot. A gomb szövegét törli. A méreteket a modul elején definiált konstansok adják

```
Const smallwd = 13 ' szélesség
Const smallht = 5  ' magasság
Const smallpd = 2  ' távolság a tartomány szélétől, padding
```

Túl kicsi cella esetén figyelmeztetést ad.

newRun Összerakja a run gombok makrójának szövegét.

newSelect Összerakja a select gombok makrójának szövegét.

setcolor(r As Range, Optional color) Ez a rutin színezi át a beillesztett elemeket a számoló munkalap aktuális színeire. Ehhez egyrészt R kód munkalap colortable nevű tartományát használja, másrészt a Rdev.xlam bővítmény ugyanilyen nevű tartományát.

Ha a color paraméter meg van adva, akkor a számoló munkafüzet color nevű színe lesz az új elem háttérszíne.

newListFillRange Kiszámolja, mi legyen az új Drop down választó űrlap Listfillrange tulajdonsága. Innen veszi az űrlap a lehetséges értékeket. Az eljárás a következő: az R kód munkalap $A oszlopának aljáról indulunk és addig megyünk, amíg nem üres mezőt találunk. Ez alá rakjuk a szükséges méretű tartományt.

6.3. Az insertR modul

Itt vannak az R kódlapok kezelésére szolgáló rutinok.

insertRfile Ennek a rutinnak a feladata az beszúrandó R file, fileok elérési útjának meghatározása. Ehhez megjelenít egy file választó dialógot, majd minden egyes kiválasztott file-ra végrehajtja a insertRcode szubrutint.

insertRcode(Rfile, wb As Workbook) A megadott Rfile tartalmát megpróbálja beszúrni a wb munkafüzetbe. Ha már létezik a megadott R kódlap, akkor rákérdez, hogy frissítse-e a (a sheetUpdate eljárással) kódlap tartalmát. Ha nem létezik ilyen nevű kódlap, akkor létrehoz egy újat és abba beilleszti a file tartalmát az insertRtoSheet eljárással.

insertRtoSheet(wb As Workbook, ws As Worksheet, Rfile) Először a sheetUpdate szubrutin segítségével feltölti a ws munkalapot a Rfile tartalmával, majd a munkalapra beilleszti a Workheet_Change esemény kezelő eljárást. Ezt arra használjuk, hogy ha a kódlap tartalma változik, akkor annak időpontját feljegyezhessük a munkalapon. A beillesztett kód

```
Private Sub Worksheet_Change(ByVal Target As Range)
  If Target.Row > 1 Then
    Me.Cells(1, 1).Value = _
      "'" & "## sheet last modified: " & CStr(VBA.Date + VBA.Time)
  End If
End Sub
```

sheetUpdate A munkalap harmadik sora után beilleszti a file tartalmát, majd az első három sorba beszúrja a fileinfo részt az insertRfileInfo eljárással.

insertRfileInfo A fileinfo a munkalap utolsó módosításának időpontja, a forrás teljes elérési útja, és a forrás utolsó módosításának időpontja. Lásd az 1.4 ábrán. Ennek beillesztése után a munkalap betű típusát Lucida console-ra állítjuk 10-es betűmérettel. Ez egy monospace betűtípus. Végül beállítjuk az első oszlop szélességét az autofit metódus segítségével.

updateAllRsources Végigmegy az aktív munkafüzet munkalapjain, és az R kódlapokat frissíti. Ha munkalap tartalma változott, rákérdez arra, hogy tényleg frissítsen-e. Egy munkalapot akkor tekintünk R kódlapnak, ha neve R kód(<filenév>) alakú. A frissítés a sheetUpdate szubrutinnal történik.

saveAllRsources Végigmegy az aktív munkafüzet munkalapjain, és az R kódlapokat menti a sheetSave eljárással. Ha ezzel információt veszhetne el, akkor a felhasználó dönthet a mentésről.

sheetSave(sh As Worksheet, Rfile) Kiírja az sh munkalap első oszlopának a tartalmát (az első három sor nélkül) a megadott Rfile állományba és frissíti a fileInfo mezőket.

getfinfo,rmprefix Ezek segédfüggvények a fileInfo mezők kezeléséhez.

6.4. Az insertSkeleton modul

Itt van az a rutin, ami a Calc.xlt template alapján létrehoz egy új számoló munkafüzetet. A modul többi része a munkafüzetek színezésével foglalkozik. Itt színekből álló collectionokkal dolgozunk, ahol az elemek kulcs-a a helyettesítendő szín RGB kódja, míg értéke az új szín RGB kódja. Ezt nevezhetnénk akár színkulcsnak is.

newSkeleton(Optional fname = False) Létrehozunk egy új munkafüzetet a Calc.xlt template alapján, beállítjuk a R.xls munkafüzetre mutató hivatkozást majd elmentjük a munkafüzetet. Ha a fname argumentum nem False, akkor ez lesz a mentett munkafüzet neve, ellenkező esetben a felhasználó választ a EXCEL saveAsFilename dialógjával. Pillanatnyilag az xls formátumot erőltetjük mentéskor. Ha ezt meg akarjuk változtatni, akkor a modul elején található wbFormat konstans értékét kell átállítani. Eredetileg ez a rutin másolta át a Thisworkbook modul checkRefToR szubrutinját az újonnan létrehozott számoló munkafüzetbe. Ennek az az előnye, hogy elegendő egy helyen a Rdev.xlam munkafüzetben javítani a kódot. Azonban, az EXCEL korábbi (XP) változata ettől rendszeresen összeomlott. Kicsit böngészve a netet arra jutottam, hogy ezt a hibát mások is tapasztalták és nem tudom javítani. Ezért került a checkRefToR a Calc.xlt sablonba is. Így azonban a javításokat mindkét helyen át kell vezetni.

repaintWb Ez a szubrutin a kijelölt tartomány első oszlopát régi színként, második oszlopát a hozzá tartozó új színként értelmezi. Az így összeállított színkulcsot alkalmazza a kijelölés munkafüzetére a changeColorsWb rutin segítségével.

changeColorsWb(wb As Workbook, colors As Collection) A wb munkafüzet minden egyes munkalapjának használt tartományára alkalmazza a changeColors rutint.

changeColors(r As Range, colors As Collection) Az r tartományt színezi át a colors színkulcs segítségével.

insertUsedColors A kijelölt tartomány celláit az őt tartalmazó munkafüzet színeivel tölti ki.

usedColors(wb As Workbook) As Collection A wb munkafüzetből kigyűjti a használt színeket.

6.5. Az insertVB modul

Ez a modul (az rDir függvény kivételével) programozottan próbál hozzáférni a EXCEL VBA projekt objektummodelljéhez. Csak akkor működik, ha a

File->Beállítások->Adatvédelmi központ->Makróbeállítások

lapon a „A VBA projekt objektmodelljéhez való hozzáférés megbízható” címkéjű checkboxot kipipáljuk.

VBAaccess Ez a függvény ellenőrzi, a VBA projekt objektummodellhez való hozzáférés lehetőségét. Ha nem engedélyezett tájékoztatja a felhasználót a teendőkről.

insertVB(ws as worksheet, code As String, procname) Beilleszti a code sztringet a ws munkalap CodeModulejába. A procname a code-ban megadott eljárás neve. Csak akkor illesztjük be a kódot, ha nincs procname nevű eljárás vagy függvény.

Ez a rutin sok fejtörést okozott, mert sokáig nem működött megbízhatóan. A megoldás kulcsa a a munkalap CodeModulejának megtalálása a getCM függvénnyel. Ha az R kódlapok beszúrása során nem jön létre a Worksheet_Change makró, akkor itt kell keresni az okát.

insertVBcoll(vbcs as collection) Erre a rutinra azért van szükség, mert az EXCEL XP változata összeomlott, ha több kódrészletet szúrtam be egymás után. A vbcs egy VBcode típusú elemekből álló collection. Minden elemnek három tulajdonsága van: id, code, ws ami egy munkalap objektum. Először megkeresi, a CodeModule-t, amibe a kódot be kell illeszteni. Ezután összefűzi azokat a kódrészleteket, amelyek idje még nincs definiálva a CodeModuleban. Végül beszúrja az így kapott szöveget.

nameExists(cl As Object, name) As Boolean Ellenőrzi, hogy a `cl` `collection` jellegű objektumban létezik-e olyan elem, aminek `name` tulajdonsága azonos a `name` paraméterrel. Tipikus használata:

```
If Not (nameExists(Wb.VBProject.References, "R")) Then
  Wb.VBProject.References.AddFromFile _
    ThisWorkbook.VBProject.References.Item("R").FullPath
End If
```

procString(cmodule as CodeModule, procname) As String Visszaadja a `cmodule` modul `procname` nevű eljárásának szövegét.

procExists(cmodule as CodeModule, procname) As Boolean Igaz értéket ad vissza, ha a `cmodule` `CodeModule`ban létezik `procname` nevű eljárás, vagy függvény.

rDir Az `appdata` környezeti változóból kiszámolja a `R.xls` munkafüzet könyvtárát.

SetReftoRxls (wb as Workbook) A `wb` munkafüzet hivatkozásai közé felvesszük a `R.xls` munkafüzetet, ha az nincs ott.

getCM(ws As Worksheet) As CodeModule A `ws` `Worksheet`hez tartozó `CodeModule`t keresi meg.

Amikor programozottan hozunk létre egy új munkalapot, akkor közvetlenül a létrehozás után a hozzá tartozó `CodeModule` elérésére

```
wb.VBProject.VBComponents ("<worksheet name>")
```

nem alkalmas. Ezért alkalmazza a rutin a körülményesebb

```
For Each vc In wb.VBProject.VBComponents
  If cname = vc.Properties("Name") Then Exit For
    Set vc = Nothing
Next
```

megoldást.

6.6. Egyebek

A `Thisworkbook` modul

A modul legnagyobb része az `EXCEL` különböző menüinek beállítása az `addMenu` rutinnal. A `removeMenu (cb, mcaption)` rutin a `cb` `commandBar` `mcaption` nevű menüjét törli. Ennek felhasználásával a `removemenus` az összes munkafüzet által telepített menüt törli.

Az `addMenu` rutinban a menük telepítése előtt a korábbi változatot töröljük. Ezt a rutint a munkalap betöltésekor és a bővítmény telepítésekor futtatjuk le. A rutin elején használjuk a `checkRefToR` függvényt. A visszatérési érték `IGAZ`, ha a megfelelő `R.xls` munkafüzetet sikerült megnyitni, vagy eleve nyitva volt és `HAMIS` egyébként. Ha van ugyan megnyitva `R.xls` munkafüzet, de elérési útja nem az elvárt, akkor megpróbáljuk a rá mutató hivatkozást törölni és a jó hivatkozást beállítani.

A menük törlését vagy a `Workbook_Close` vagy a `Workbook_AddInUninstall` rutinból indítjuk el. A munkafüzet bezárásakor csak akkor töröljük a menüket, ha a `Rdev.xlam` munkafüzet nincs bővítményként telepítve.

A `capturewnd` modul

A `capturewnd` szubrutin a megadott ablak (`hwnd`) képét `bmp` formátumban menti a megadott fileba (`fname`). Ennek a dokumentációnak a képernyőképei készültek a segítségével.

A `saveVBE` modul

Egyetlen szubrutint definiál `saveAllVBE` névvel. A felhasználó által kiválasztott könyvtárba menti a megnyitott munkafüzetek `Visual Basic` kódját. Mindegyik `VBAProject`-et a nevével megegyező alkönyvtárba próbál menteni. Mentés előtt rákérdez, hogy az adott projektet valóban menteni szeretnénk-e. Az egyes `VB` komponensek mentése az export metódussal történik.

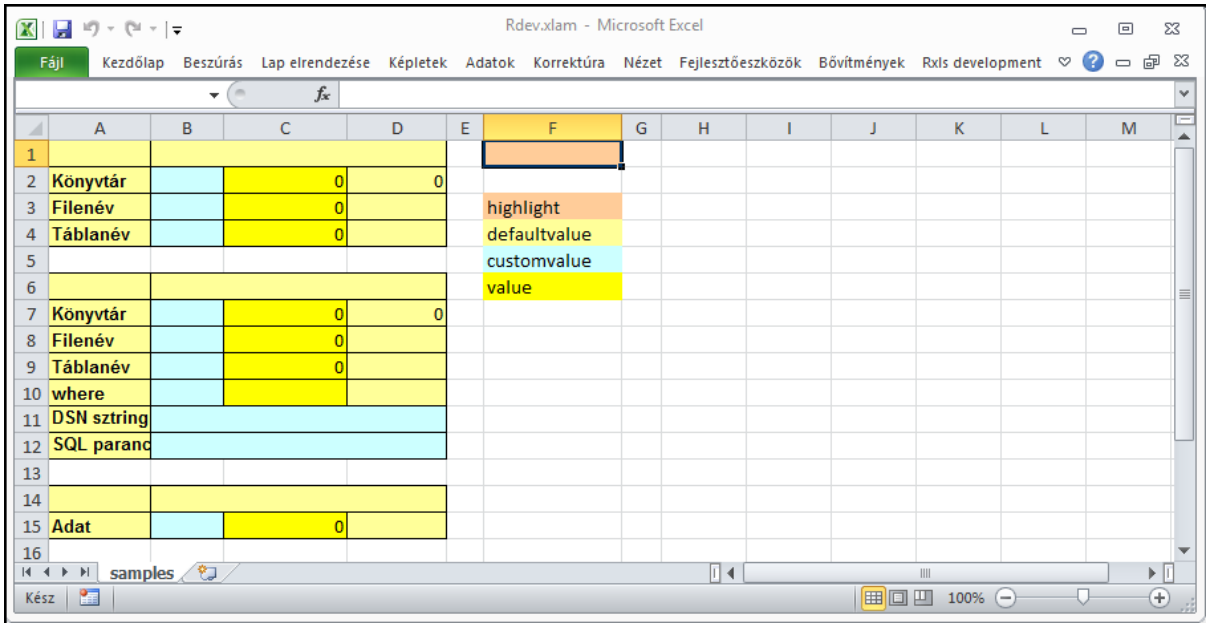
A `RIBBON` modul

Egyetlen szubrutint tartalmaz, `menufunction` névvel. Ez a `RIBBON` felületen definiált gombok makrója, a gomb azonosítója alapján a megfelelő eljárást hívja meg.

Azért, hogy a bővítményt `EXCEL` XP-be is be lehessen tölteni ez a modul csak akkor fordítódik le, ha `VBA7` konstans definiálva van. Ez biztosan teljesül `OFFICE 2010` és 2013 esetén. Az `OFFICE 2003` ill. `OFFICE 2007` változatokkal nincs tapasztalatom.

A samples munkalap (6.1 ábra)

Ezen a munkalapon vannak az `Access Form` és `ACCESS + SQL` valamint a `Data Line` elemek mintái. Innen másoljuk át őket a megfelelő menüparancs hatására.



6.1. ábra. A samples munkalap képe

7 Az R oldal részletesebben

7.1. Az Rxls csomag rutinjai

XLdata

Az XLdata függvény célja, hogy viszonylag könnyen lehessen a beállításokat, paramétereket az R-nek átadni. A függvény argumentumai:

```
XLdata
function (c1 = 1, c2 = 2, c2.offset = c2 - c1, range, onlyvisible = TRUE,
    rmHidden = FALSE, env = parent.frame(), XLrange = {
        on.exit(XLrange <- NULL)
        if (missing(range))
            range <- XL$selection()$address(external = TRUE)
        .XLrange(range, XL = XL)
    }, XL = THISXL, value = "value2")
```

c1,c2,c2.offset Az első két paraméter a neveket ill. értékeket tartalmazó oszlopok sorszáma a tartományon belül, a harmadik ezek különbsége. A kód a c2.offset értéket használja, így a c2 értéket csak akkor használjuk, ha c2.offset nincs megadva.

range A tartomány címe, lehetőleg teljes név, azaz munkafüzet, munkalap és azon belül a tartomány bal felső és jobb alsó cellája. pl. [Munkafüzet1]Munka1!$A:$F. Az extAddress munkalapfüggvénnyel könnyen megadható.

rmHidden Ha TRUE, akkor a rejtett sorokban lévő változókat töröljük az R oldalon. Alapértelmezésben nem törlünk.

onlyvisible Ha TRUE (ez az alapértelmezés), akkor csak a látható sorokat használja, ellenkező esetben a rejtetteket is.

env Az környezet (environment), amiben a változók definiálva lesznek. Alapértelmezésben az a függvény hívásakor aktív környezet. Ha a background(eval(.FUN)) vagy eval(.FUN) részeként használjuk az XLdata függvényt, akkor a globális .GlobalEnv környezetben lesznek definiálva a változók. Tipikusan ezt a paramétert nem kell megadni.

XLrange A tartomány címének megadása helyett egy EXCEL range-t reprezentáló COM objektumot is megadhatunk. Tipikusan ezt a paramétert nem kell megadni.

XL A használandó EXCEL példány. Tipikusan ezt a paramétert nem kell megadni.

value A cella melyik érték tulajdonságát használjuk? "value" vagy "value2". Csak a dátum/idő adatok kezelésében van különbség. "value" esetében az EXCEL dátum/idő adatai POSIXt típusú adatként jelennek meg az R oldalon, míg "value2" esetében az

1900.01.01. dátum óta eltelt napok számaként. A POSIXt típus konverziót a téli/nyári időszámítás összezavarja, ezért inkább a másikat használtam mindig. "value2" használata esetén az eredmény az XLDate függvény segítségével Date típusúvá alakítható. Tipikusan ezt a paramétert nem kell megadni.

XLsource

Ez a függvény a source mintájára működik, azaz a kódot olvas be szövegként, majd azt elemzi (parse) és kiértékeli, vagyis végrehajtja. Azonban a source-től eltérően az XLsource nem file-ból, hanem egy EXCEL munkafüzet R kódlapjáról olvassa be a kód szövegét. Az argumentumok:

```
XLsource
function (..., wbname = .wbname, name = {
    dots <- match.call(expand.dots = FALSE)$...
    if (length(dots) && !all(sapply(dots, function(x) is.symbol(x) ||
        is.character(x))))
        stop("... must contain names or character strings")
    sapply(dots, as.character)
}, env = parent.frame(), wsname = sprintf("R kód(%s)", name),
    wb = XL[["workbooks", wbname]], XL = THISXL)
```

... Az R kódlap-ok nevei, *nem szükséges idézőjelek közé tenni a neveket*. Ha a nevek karakter vektor formájában állnak rendelkezésre, akkor használjuk a name argumentumot. Ezek a nevek nem a munkalap nevek. Azok R kód(kódlap név) alakúak, lásd a wsname argumentumot.

wbname A munkafüzet neve. Ha nem adjuk meg, akkor az aktív munkafüzetet fogja a rutin használni.

name A kódlapok neveit c ("<név1>","<név2>",...) alakban is megadhatjuk. Tipikusan ezt az argumentumot nem kell megadni.

env Hasonlóan az XLdata függvényhez itt is megadható, hogy melyik környezetben dolgozunk. Ugyanaz a mechanizmus érvényesül itt is. Tipikusan ezt az argumentumot nem kell megadni.

wsname Az R kódot tartalmazó munkalapok nevei. Tipikusan ezt az argumentumot nem kell megadni.

wb A munkafüzetet reprezentáló COM objektum. Tipikusan ezt az argumentumot nem kell megadni.

XL A használni kívánt EXCEL példány. Tipikusan ezt az argumentumot nem kell megadni.

XLwritedf

Egy data.frame-t másol EXCEL munkafüzetbe. Külön figyelni kell a factor és dátum típusú adatokra. Lehetőség van az oszlop szélesség tartalomhoz igazítására és név megadására.

```
XLwritedf
function (XLrange, df, with.names = TRUE, autoFit = TRUE,
    setname)
```

XLrange A céltartomány. A df data.frame-t az XLrange tartomány bal felső sarkába illesztjük be.

df A másolandó data.frame.

with.names Ha TRUE, akkor a kitöltött tartomány első sorába a data.frame nevei kerülnek.

autoFit Ha TRUE, akkor az oszlopok szélességét a tartalomhoz igazítjuk.

setname Ha megadjuk és értéke karakterlánc, akkor ez lesz a kitöltött tartomány (munkafüzet szintű) neve.

XLreaddf.cols, XLreaddf.rows

Mindkét függvény az EXCEL megadott tartományát alakítja R data.frame-é. A különbség abban van, hogy míg a XLreaddf.cols esetén a tartomány oszlopai felelnek meg az eredmény data.frame oszlopainak, addig az XLreaddf.rows esetében a tartomány soraiból lesznek a data.frame oszlopai. Az előfeldolgozás (beolvasás, összevont cellák ellenőrzése) után mindkét rutin az XLlist2df függvényt használja és ... argumentumot tovább adja ennek a függvénynek. Így a paraméterek listája hosszabb, mint ami az alábbi fejlécből következne.

```
XLreaddf.cols
function (range, which.cols = seq_len(ncol(x)), XLrange = {
    on.exit(XLrange <- NULL, add = TRUE)
    if (missing(range))
        XL$selection()
    else .XLrange(range, XL = XL)
}, ..., chk.merge = TRUE, value = "value2", XL = THISXL)
XLreaddf.rows
function (range, which = seq_len(nrow(x)), XLrange = {
    on.exit(XLrange <- NULL, add = TRUE)
    if (missing(range))
        XL$selection()
    else .XLrange(range, XL = XL)
}, ..., chk.merge = TRUE, value = "value2", XL = THISXL)
```

range A céltartomány szövegesen megadott címe. Ha nem adjuk meg, akkor az aktuális kijelölést használjuk.

which.cols Itt lehet megadni, hogy a tartomány mely oszlopai szerepeljenek a végeredményben. Alapértelmezése a tartomány összes oszlopa.

XLrange A tartomány COM objektumként. Jellemzően nem kell megadni.

... Ezeket a paramétereket tovább adjuk a XLlist2df függvénynek.

```
Rxls:::XLlist2df
function (x, header = 1, trim = FALSE, skip = 0, head.sep = "",
    numeric = FALSE, ...)
```

Az így keletkező további beállítási lehetőségek:

header A tartomány fejlécének nagysága, sorokban.

trim Ha igaz, akkor az üres sorokat és oszlopokat eldobjuk, ha hamis nem történik semmi.

skip Lehet függvény, ekkor a trimmelés után ezt alkalmazzuk az adatokra. Lehet szám is. Pozitív érték esetén skip számú sort eldobunk.

head.sep Akkor használjuk, ha a tartomány fejléce több soros. Ilyenkor a data.frame neveit az oszlop fejlécében szereplő sztringek összefűzésével kapjuk.

numeric Ha igaz, akkor a data.frame elemeit megpróbáljuk numerikus értékre konvertálni akkor is, ha mondjuk az EXCEL-ben szövegesen szerepelnek a számok. A tizedesjel az R tizedes jele, alapértelmezésben ez a pont.

... Ezeket a paramétereket továbbadjuk a data.frame függvénynek. Célszerű lehet a stringAsFactors = FALSE megadása. Ugyanis szöveges adatból data.frame létrehozása során factor változó lesz. Ha mégis inkább szöveges adatot szeretnénk a data.frame-ünkben, akkor ezt a beállítást kell használni.

Először trimmelünk ha kell, majd a skipet alkalmazzuk, ezután választjuk le a fejlécet a header alapján. A megmaradó adatokra alkalmazzuk a as.numeric függvényt, ha szükséges, majd a data.frame függvényt. Végül beállítjuk a data.frame neveit, ha vannak.

chk.merge Ha a tartományban összevont cellák vannak, akkor a képernyőn látható érték az összevont cellák bal felső sarkában lévő cella értéke, a többi cella értéke üres. Ez a R oldalon NA értéket eredményez. Ha chk.merge értéke TRUE, akkor az R oldalra másolt értékek megegyeznek a cellában látható értékkel. Vagyis egy összevont tartomány minden cellájához a képernyőn látható értéket rendeljük.

value Melyik tulajdonságát használjuk a tartománynak. Tipikusan nem kell megadni. Lehet "value", "value2", "formula" vagy akár "formulalocal".

XL Melyik EXCEL példánnyal dolgozzunk. Tipikusan nem kell megadni. Alapértelmezésben a globális THISXL érték.

XLget

Egy EXCEL tartományt olvas be R-be, pontosabban a tartomány és az adott munkalap usedrangeének metszetét.

```
XLget
function (range, XLrange = {
    if (missing(range))
        XL$selection()
    else .XLrange(range, XL = XL)
}, value = "value", XL = THISXL)
```

range A tartomány címe, sztring. Lehet az aktív munkafüzetben definiált név is. Ha nem adjuk meg, akkor a kijelölt tartomány.

XLrange Az olvasandó tartományt reprezentáló COM objektum. Tipikusan nem kell megadni.

value Melyik tulajdonságát használjuk a tartománynak. Tipikusan nem kell megadni. Lehet "value", "value2", "formula" vagy akár "formulalocal".

XL Melyik EXCEL példánnyal dolgozzunk. Tipikusan nem kell megadni. Alapértelmezésben a globális THISXL érték.

ACCESS.get

ACCESS adatbázisból olvas be egy táblát.

```
ACCESS.get
function (dir, file, table, where = "", fields = "*")
```

dir Az adatbázis könyvtára.

file Az adatbázis fileneve.

table A tábla neve.

where A lekérdezés where része. Ha nem adjuk meg a teljes táblát lekérdezzük.

ODBC.get

ODBC adatbázisból olvas be egy táblát.

```
ODBC.get
function (dsn, sql)
```

dsn A kapcsolatot leíró sztring.

sql Az lekérdezés sql parancsa.

getFromDB

Választ a két előbbi rutin közül. Ha a dsn argumentumként nem definiált változót adunk át, akkor az ACCESS.get egyébként a ODBC.get függvényt használjuk.

```
getFromDB
function (dir, file, table, where = "", dsn, sql)
```

logDevice kezelés

logDev...

A logDev függvény új logDevice-t hoz létre a megadott típussal. A logDev.set függvény az aktuális logDevice-t állítja át, logDev.off kikapcsolja azt, míg a logDev.list kilistázza a létező logDevice-okat.

```
logDev
function (...)
logDev.set
function (num)
```

... Az új logDevice típusa. "null", "R", "EXCEL", "TCL" lehet.

num Az új aktív eszköz sorszáma.

progress

Új progressbart ad az aktív logDevicehoz.

```
progress
function (...)
```

A paramétereket továbbadjuk. A tényleges argumentumok:

pattern Az sprintf függvény által használt formátum sztring.

... A formátum sztringbe helyettesítendő kezdő értékek.

lazyness Numerikus érték, a progressbar frissítési időköze másodpercben.

A progress visszatérési értéke egy objektum (list) a következő függvényekkel:

inc() Ha progressbar mintájának egyetlen numerikus argumentuma van, akkor azt egyel növeli és frissíti a logDevicet a lazyness paraméterrel összhangban.

step(...) A progressbar értékét frissíti a sprintf(pattern,...) függvényhívással és frissíti a logDevicet a lazyness paraméterrel összhangban.

hide() Ideiglenesen leveszi a progressbar-t a logDevice-ról. Nem frissít!

show() Visszateszi a progressbar-t a logDevice-ra. Nem frissít!

reset(pattern,...,lazyness) A progressbar paramétereit átállítja. Nem frissít!

refresh() Frissíti a logDevice szövegét, függetlenül az előző frissítés óta eltelt időtől.

rm() Véglegesen törli a progressbart a logDevice-ról. Nem frissít.

logMessage

```
logMessage
function (..., sleep = 0)
```

... Az sprintf függvénynek átadott paraméterek, az első elem lehet formátum sztring.

sleep Várakozás a message megjelenítése után másodpercben.

Dátum kezelő függvények

Az EXCEL-ből "value2" tulajdonságon keresztül kapott numerikus értéket az R valamelyik dátum típusára konvertáló függvények.

```
XLDate
function (date, d1904 = FALSE)
XLPOSIXct
function (date, d1904 = FALSE)
```

date Numerikus érték. A kezdő időpont óta eltelt napok száma. Nem feltétlenül egész szám.

d1904 Kezdő időpont. Az EXCEL munkafüzetek Date1904 nevű tulajdonságának felel meg.

Egyebek

XLScreenUpdateTurnOff

Kikapcsolja az EXCEL ScreenUpdate tulajdonságát és a Calculation tulajdonság értékét az xlCalculationManual értékre állítja. A visszatérési érték egy függvény, ami a visszaállítja az eredeti állapotot. Lehetőség van az aktív cella értékének visszaállítására is.

```
XLScreenUpdateTurnOff
function (Appl, activecell = FALSE)
```

Appl Az az EXCEL application, aminek az képernyő frissítését le akarjuk tiltani.

activecell Ha igaz, akkor az aktív cella helyét is megjegyezzük és a helyreállítás során a cellát aktiváljuk. Csak akkor állítsuk igazra, ha időközben nem töröljük a cella munkalapját vagy munkafüzetét.

XLusedrange

Visszaadja egy tartománynak és a usedrange tartománynak a metszetét. Az eredmény egy range objektum.

```
XLusedrange
function (r)
```

r Az a range objektum, (tehát nem szöveges cím) amit a usedrange tartománnyal metszeni akarunk.

pkgzip

win-binary formátumban ment telepített R csomagokat.

```
pkgzip
function (pkgs = {
    preselect <- grep("(com(|proxy)|Rxls)$", rownames(ipkgs),
        value = TRUE)
    if (ask)
        select.list(rownames(ipkgs), preselect, multiple = TRUE,
            title = "Zippelendő csomagok:", graphics = T)
    else preselect
}, zip.mappa = choose.dir(Sys.getenv("HOME"), "Zip állományok helye:"),
    ask = interactive(), PACKAGES = FALSE)
```

pkgs Az R csomagok listája. Ha nem adjuk meg és az ask paraméter igaz, akkor interaktív módon választhatunk a telepített csomag listájából. Ha hiányzik és az ask paraméter hamis, akkor értéke c("com","comproxy","Rxls").

zip.mappa A zippelt csomagok helye. Ha nem adjuk meg, akkor egy könyvtár-választó dialóg segítségével jelölhetjük ki.

ask Megkérdezzük-e a felhasználót?

installRxls

Az Rxls csomagban lévő EXCEL fileokat átmásolja a <appdata>/Rxls könyvtárba és a verzió információkat a versions nevű fileba írja.

```
installRxls
function ()
```

R_RegEntries

Az R-rel kapcsolatos regisztry bejegyzésket listázza ki. Ezek egy része a Software/R-core kulcs alatt található. Egy másik része a com csomag által bejegyzett CLSID és Typelib adat.

```
R_RegEntries
function ()
```

7.2. VBA kód konvertálása R kóddá

Szükség lehet arra, hogy R-ben írjunk meg olyan programrészeket, amik EXCEL VBA-ban már megvannak, vagy ott könnyen előállíthatóak, mondjuk a makrórögzítés funkció segítségével. A konverzió nem túl komplikált. A konstansokat kigyűjthetjük a Visual Basic szerkesztő Object browser-éből. Ahol EXCEL objektumot használunk (Selection, ActiveCell, ActiveSheet, stb) ott valójában az Application objektum tulajdonságaira hívatkozunk, tehát a teljes leírás az VBA-ban Application.Selection lenne és hasonlóan a többi esetben is. Visual Basic-ben a . (pont) operátor egy objektum tulajdonságát vagy metódusát kérdezi le. Erre az R oldalon a "[[", ill. $ operátorok használhatóak. Az elsővel tulajdonságot, a másodikkal metódust vagy tulajdonságot érhetünk el. Azaz az EXCEL workbooks collection-ját kétféleképpen is elérhetjük.

```
> THISXL[["workbooks"]]
<pointer: 0x0259534c>
attr(,"class")
[1] "COMObject"
> THISXL$workbooks()
<pointer: 0x0259597c>
attr(,"class")
[1] "COMObject"
```

Ha az első munkafüzetet akarjuk elérni, akkor a Workbooks objektum item tulajdonságát kellene használnunk. Ez a default tulajdonság, tehát a következők valamennyien használhatóak

```
THISXL[["workbooks", 1]]
THISXL[["workbooks"]][["item", 1]]
THISXL$workbooks(1)
THISXL$workbooks()$item(1)
THISXL[["workbooks"]]$item(1)
THISXL$workbooks()[["item", 1]]
```

Ha metódust akarunk meghívni, akkor csak a $ szintaxis használható

```
> wb <- THISXL$workbooks()$add()
```

Ha írható tulajdonságot akarunk beállítani, akkor csak a [[szintaxis használható értékadással kombinálva. A következők mind az ActiveCell tartomány értékét állítják 1-re.

```
> withObj(THISXL,.[["activecell"]]<-1)
> withObj(THISXL[["activecell"]],.[["value"]]<-1)
> THISXL[["activecell"]] <- 1
> THISXL[["activecell"]][["value"]] <- 1
> ac <- THISXL$activecell()
> ac[["value"]] <- 1
> ac <- NULL
```

A segédváltozós megoldásban az ac<-1 nem működne, egyszerűen 1 értéket adnánk az ac nevű változónak.

Miután létrehoztunk egy hivatkozást valamely COM objektumra, használat után mindig töröljük azt. Ha egy függvény belsejében tesszük ezt akkor a hivatkozás törlése automatikusan megtörténik a függvényből való kilépés után (pontosabban az azt követő első garbage collection alkalmával). Globális szintű változók használata esetén kézzel kell a változó értékét NULL-ra változtatni, vagy az rm függvénnyel törölni a változót magát.

```
> wb$close(FALSE)
[1] TRUE
> wb <- NULL
> gc()
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 574199 15.4     940480 25.2   940480 25.2
Vcells 653940  5.0    1308461 10.0  1237154  9.5
```

Előfordulhat, hogy nem elég törölni a hivatkozást az adott objektumra, pl. mert egy új ACCESS, EXCEL példányt vagy új munkafüzetet hoztunk létre, amire a továbbiakban nincs szükség. Ha ez egy függvény belsejében történt, akkor jól használható az R on.exit lehetősége. Ezzel olyan kódot tudunk megadni, amit a függvény kiértékelése után hajt végre az R. A következő kódrészlet a függvény elején létrehoz egy új munkafüzetet majd kilépéskor mentés nélkül bezárja.

```
f <- function() {
  wb <- THISXL$workbooks()$add()
  on.exit(wb$close(FALSE), add = TRUE)
  ## ...
}
```

Egy meglepő jelenség

Időnként nem pontosan az történik, amit várunk. Tegyük fel, pl. hogy az activeCell cellát szeretnénk az =MA() formulával ellátni. Kézenfekvő megoldásnak tűnik a következő sor

```
> THISXL[["activecell"]][["formula"]] <- "=MA()"
```

Azonban ennek hatására a cella értéke ugyan az aktuális dátum lesz, de a formula nélkül.

```
> THISXL[["activecell"]][["formula"]]
[1] "42252"
> XLDate(THISXL[["activecell"]][["value2"]])
[1] "2015-09-05"
```

Ha azonban beiktatunk egy közbülső lépést, akkor a várt eredményt kapjuk.

```
> withObj(THISXL[["activecell"]], .[["formula"]] <- "=MA()")
> THISXL[["activecell"]][["formula"]]
[1] "=MA()"
> XLDate(THISXL[["activecell"]][["value2"]])
[1] "2015-09-05"
```

A különös viselkedés megértéséhez kiírjuk a végrehajtott lépések részleteit.

```
> `[[.COMObject` <- function(handle, property, ...) {
+   print(match.call())
+   cat("\thandle =", deparse(handle), "\n")
+   x <- com::`[[.COMObject`(handle, property, ...)
+   cat("\treturn value =", deparse(x), "\n\n")
+   x
+ }
> `[[<-.COMObject` <- function(handle, property, ...) {
+   print(match.call())
+   cat("\thandle =", deparse(handle), "\n")
+   x <- com::`[[<-.COMObject`(handle, property, ...)
+   cat("\treturn value =", deparse(x), "\n\n")
+   x
+ }
> THISXL[["activecell"]][["formula"]] <- "=MA()"
`[[.COMObject`(handle = `*tmp*`, property = "activecell")
handle = <pointer: 0x02595424>
return value = <pointer: 0x02594fec>

`[[<-.COMObject`(handle = `*tmp*`, property = "formula",
    value = "=MA()")
handle = <pointer: 0x02594fec>
return value = <pointer: 0x02594fec>

`[[<-.COMObject`(handle = `*tmp*`, property = "activecell",
    value = <pointer: 0x02594fec>)
handle = <pointer: 0x02595424>
return value = <pointer: 0x02595424>
```

Azaz az történt, hogy először a `THISXL` objektum `activeCell` tulajdonságát kérdeztük le, majd az így kapott `COM` objektum formula tulajdonságát beállítottuk. Ezután a változásokat „átvezettük” a `THISXL` objektumra is. Pontosabban az utolsó lépésben `THISXL` `activecell`-jét a második lépésben kapott eredményre állítjuk, ami ugyanez az aktív cella. Csakhogy ez azt eredményezi, hogy mindkét oldalon a `COM` objektum `default` tulajdonságát használjuk, vagyis valójában az `activecell` `value` tulajdonságát állítjuk saját magára. Visual Basicben kifejezve az alábbi kódnak megfelelő lépések történnek:

```
set tmp = Application.activeCell
tmp.formula = "=MA()"
Application.activeCell = tmp
```

Ahol az utolsó sor azzal ekvivalens, hogy

```
Application.activeCell.value = tmp.value
```

Ez az utolsó lépés az, ami a formulát eltünteti és helyettesíti az aktuális értékkel. Amikor a második változatot használtuk, ahol egy segédváltozóban tároltuk az `ActiveCell` objektumot, erre a lépésre nem került sor.

Összefoglalva, ha valamilyen tulajdonságot állítunk be a `COM` interfész segítségével, célszerű a segédváltozókat használni, hogy a fenti példában látott mellékhatást elkerüljük.

Az `R language definition` leírás 3.4.4 `Subset assignment` című részében részletesen olvashatunk az értékadó függvényekről.

7.3. Dokumentáció, `vignette`-ák

Valamennyi általam készített R csomag esetén a kommentezett kód böngészhető pdf formátumban. Pl. az `Rxls` csomag esetén a következő utasítást kell kiadni R-ben.

```
> vignette("Rxls")
```

A megjelenítéshez szükséges, hogy a file társítások között regisztrálva legyen pdf nézegető program, pl. Acrobat Reader.

`RStudio` használata esetén, mind az `R` konzolban, mind a szövegszerkesztő ablakban elérhető a kód „kiegészítése” funkció a `TAB` billentyű lenyomásával. További előnye az `RStudio` használatának, hogy a változó nevek kiegészítése, mellett azok rövid leírása is megjelenik `tooltip`-szerűen.

Ha egy adott `R` függvény nevére nem emlékszünk pontosan, akkor jól használható a `apropos` függvény, ami a megadott mintára illeszkedő függvényneveket sorolja fel.

8 Hibaelhárítás

8.1. COM problémák, DebugView

Előfordulhat, hogy egy látszólag hibátlan COM interfészt használó hívás esetén nem az történik amit várunk, a legtöbb ilyen esetben a visszatérési érték NULL. A com csomag nem ír ki hiba üzeneteket az R konzolra. Üzeneteit a DebugView programmal lehet megnézni. Ezt a programot a Microsoft honlapjáról lehet letölteni:

http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx.

Futtatásához rendszergazda jogosultság kell.

8.2. R kód debugolása

Ha az R kódunk hibát (error), vagy akár csak figyelmeztetést (warning) generál, akkor célszerű megkeresni az okát. A legelső eszköz amit itt használhatunk a traceback függvény, ami a hiba helyének behatárolásához ad segítséget. Ha a hibát generáló függvény megvan, akkor használhatjuk a debug függvényt. Hatására az argumentumában megadott függvény végrehajtását lépésenként tudjuk nyomon követni.

Ha egy megadott függvény adott pontjától szeretnénk ugyanezt tenni, akkor a kódba ideiglenes beszúrhatjuk a browser() függvényhívást. Részletesebben lásd az R help-jében, a Writing R Extensions leírás 4., Debugging fejezetében. Ugyancsak olvashatunk az R debugolásáról az R language definition leírás 9. fejezetében.

Emellett célszerű lehet kipróbálni a debug csomagot. Ez hasonló felületet nyújt, mint az EXCEL Visual Basic szerkesztője, azaz a debugolt kód külön ablakban jelenik meg, és színek jelzik, hogy hol tart a kiértékelés.

Az RStudio környezet a 0.98-as verziótól kezdve szintén tartalmaz hibakeresési támogatást.

9 Teendők R verzió váltásnál

9.1. R csomagok installálása forrásból, type="source"

Windows alatt a csomagokat jellemzően zip file-ból telepíti az ember. Ennek legfőbb előnye, hogy nem szükséges fordító programot telepíteni a gépre. Másfelől, mégis célszerű lehet forrásból telepíteni, abban az esetben, ha a csomag zip változata valamilyen ok miatt nincs fenn a CRAN-on. Ha ragaszkodunk a menü használatához és mégis forrásból akarunk telepíteni, akkor célszerű a pkgType opciót átállítani "both"-ra. Ez megtehető az alábbi módon.

```
> options(pkgType = "both")
```

Ha a csomag forrása az esetleg szükséges Rtools-zal együtt megtalálható a gépünkön, akkor az R konzolból a következő képpen indíthatjuk a telepítést:

```
> install.packages("<csomag elérési útja>", type = "source", repos = NULL)
```

Ha a csomagban nincs fordítandó kód (C/C++, vagy Fortran), akkor a fenti módszer az Rtools telepítése nélkül is működik.

Az Rtools (Windows toolset) készlet letölthető az R projekt honlapjáról:

http://CRAN.R-project.org/bin/windows/Rtools/.

A fordító készletről az előbbi linken ill. az „R Installation and Administration" című leírásban lehet további részleteket találni. Ez utóbbi az R help-jéből elérhető pdf, vagy html formátumban.

com, comproxy fordítása

A com csomag, jelenleg csak 32-bites módban fordítható. Ezért a telepítésekor, vagy frissítésekor a INSTALL_opts argumentumra is szükség van.

```
> install.packages("<com tar.gz elérési útja>", type = "source", repos = NULL,
INSTALL_opts = "--no-multiarch")
```

A com csomagnál az utolsó argumentum nélkül is végigmegy a telepítés, mert csomag forrásában nem üres a configure.win file van. Nekem úgy tűnik, hogy ez egy nem dokumentált viselkedés, vagyis a későbbi verziókban változhat.

Ha az utolsó INSTALL_opts = "-no-multiarch" argumentumot is megadjuk, akkor az R csak a fő architektúrára, vagyis 32-bites R esetén csak 32-bites módra próbálja fordítani a csomagot. Mivel az Rxls csomag a com csomagra épül, annak telepítése során is szükség van erre az argumentumra. Ez a csomag nem tartalmaz fordított kódot, így egy nem üres configure.win file elhelyezése nem segít.

com és comproxy 64 bites változata

A com csomag 1.0-68 ill. a comproxy 1.0-16-os verziójától kezdve mindkét csomag használható 64 bites R-rel is és a telepítés során az INSTALL_opts="-no-multiarch" opcióra nincs szükség. Azaz a forrásból történő telepítéshez a következő R parancs használható:

```
> install.packages("<com tar.gz elérési útja>", type = "source", repos = NULL)
```

A módosítás során, a néhány helyen még előforduló, assert(...) hívásokat lecseréltem. Az assert(...) hivások tipikusan olyan ellenőrzések voltak, amik normál működés mellett nem okozhattak gondot, viszont ha mégis, akkor a futó R példány összeomlását eredményezték.

További változás com_1.0-68-ban, a korábbi változathoz képest, hogy a rcom_srv.tlb és StatConnLib.tlb típus könyvtárak az R telepítés

library/com/lib/i386

mappájából a

library/com/binary

mappába kerültek át. Ezt a változtatást szintén a 64 bites változat tette szükségessé. Korábban a típus könyvtár betöltése a com.dll mappájából történt, ami viszont attól függ, hogy 32 vagy 64 bites R-et használunk-e. A registry-ben ezzel szemben egy helyet definiálhatunk a típus könyvtárak helyének ugyanis a bitek számától függetlenül ugyanazt az interface-t implementálja mind a 32, mind a 64 bites DLL.

A típus könyvtár helyének változása miatt a frissítés után a comRegisterRegistry() R függvényt le kell futtatni egy rendszergazdai jogosultságokkal inditott R példányban, lásd az 1.1 ábrán. A telepítő ezt a lépést automatikusan elvégzi, így ennek használatakor erre a lépésre nincs szükség.

9.2. Egyebek

Van olyan korábban fejlesztett alkalmazás, ami ACCESS adatbázisból adatokat másol adotokat EXCEL munkalapra. Ehhez a DAO3.6 library-t használtam. Ez nem érhető el a Windows újabb változataiban. Az interneten keresgélve azt találtam, hogy a jelenlegi megfelelője:

Microsoft Office 14.0 Access database engine Object Library

Ha ez a hiba előfordul, akkor a szóbanforgó munkafüzet (pl. az arviz csomag Karkalkulator munkafüzetében) a hivatkozást (VB editor->References menüpontját használva) le kell cserélni.

10 Újdonságok az Rxls csomagban az 1.0-119 verzióhoz képest

Amikor ezt a leírást készítettem, tapasztalnom kellett, hogy a korábbi változat nem működik a 64 bites EXCEL 2010 programmal. Ezt pár apró változtatás követte, majd újra átgondoltam néhány megoldást, amit újabb változtatások követtek. Ezért úgy döntöttem, hogy a verzió számot 2.0-0-ra növelem, és mindent átírok, amiről úgy gondolom jobban, egyszerűbben is megoldható.

Rxls könyvtár

A korábban készített munkafüzeteknél az egyik gond a hordozhatóság volt. Minden ilyen (számoló) munkafüzet hivatkozást tartalmazott az R.xls filera. Ez tartalmazza a szükséges Visual Basic kód nagy részét. A gondot az okozza, hogy amikor egy másik gépre másoljuk a számoló munkafüzetet, vagy verzió váltás miatt a korábbi, R.xls könyvtár megszűnik, akkor hivatkozás elromlik, és azt kézzel kellett javítani. Erre két megoldás látszik. Az egyik, hogy az R.xls állományt az R könyvtár struktúráján kívül tároljuk, a felhasználó meghatározott könyvtárában, pl. {APPDATA}/Rxls/R.xls. A hivatkozó munkafüzet tudja a relatív helyet az {APPDATA} könyvtárhoz képest, amit pedig ki tud olvasni egy környezeti változóból. A számoló munkafüzet megnyitáskor ellenőrzi, hogy a hivatkozott R.xls állomány meg van-e nyitva, ha nincs megkísérli a standard helyről betölteni. Ennek eredményeként a számoló munkafüzet hordozhatóvá válik; ha átvisszük egyik gépről a másikra, vagy verzió váltás történik, továbbra is használható marad feltéve, hogy az Rxls csomag megfelelően van telepítve.

Ez tehát két ponton jelent változást: egyrészt minden számoló munkafüzetnek tartalmaznia kell a megfelelő Workbook_open eljárást, másrészt az R oldalon Rxls csomag betöltésekor az R.xls állományt át kell másolni a szokásos helyére, ha az nincs ott. Ha forrásból telepítjük a csomagot, akkor az átmásolás a telepítés részeként is történhet.

Hasonló gond jelentkezhet az RCOM 1.0 Type Library-val kapcsolatban is. Itt a megoldás az, hogy telepítés után rendszergazdaként kell elindítani az R programot és ki kell adni a következő utasítást:

```
> comRegisterRegistry()
```

A másik lehetőség a hordozhatóság elérésére az lehet, hogy a véglegesítés során a szükséges modulokat bemásoljuk a számoló munkafüzetbe.

10.1. com csomag

Ennek az útmutatónak az írása közben vettem észre, hogy tulajdonságok lekérdezésénél csak pozicionális argumentum átadásra van lehetőség, név szerintire nem. Ez a 1.0-49 verziótól

kezdve annyiban módosul, hogy tulajdonságot kétféleképpen is le lehet kérdezni. A régi szintaxis

```
<obj>[["<tulajdonság név>", <argumentumok>]],
```

továbbra is használható. Ekkor azonban hiába vannak az argumentumok névvel ellátva, azokat pozíciójuk szerint adjuk át. Korábban ez volt a viselkedés.

Emellett a metódusokra megszokott szintaxis is használható

```
<obj>$<tulajdonság név>(<arg1>=<value1>,...).
```

Ez utóbbi a `<obj>.<tulajdonság név> <arg1>:=<value1>, ...` Visual Basic szintaxisnak felel meg. Ugyanúgy, ahogy Visual Basic-ben itt sem szükséges neveket megadni, viszont a zárójel akkor is kell, ha nincs argumentum.

comCheckRegistry

`TRUE` értéket ad vissza, ha a `Registry` tartalmazza a `RCOM` szerverre mutató bejegyzést ellenkező esetben az érték `FALSE`. Arra szolgál, hogy a telepítés állapotát ellenőrizni lehessen.

```
> comCheckRegistry()
[1] TRUE
```

Egészen pontosan a következők ellenőrzése történik meg: regisztrálva van-e az `R COM` szervere, regisztrálva van-e a `COM` szerver `type library`-ja, végül az `R` legfrissebb változata alól van-e regisztrálva. Ha valamelyik pont esetében a regisztry bejegyzés nem létezik, vagy nem az, aminek lennie kellene a felhasználó figyelmeztetést kap a teendőkkel együtt.

RODBC

A `CRAN`-ról letölthető `RODBC` csomag `odbcConnect` függvénye másképp viselkedik `RGui`-ból használva, mint egyébként. Ha `Rgui`-ból indítjuk paraméterek nélkül, akkor egy felugró ablakból kiválaszthatjuk a használandó adatbázist, megadhatjuk a felhasználó nevet és a jelszót, ha ez szükséges, stb. Egyéb formáját használva az `R`-nek ugyanez az eljárás hibát eredményezne.

A munka során átadtam egy patchelt változatot az `RODBC` csomagból, ami ezt a különös viselkedést orvosolja. A javasolt változtatást a csomag karbantartójának is jeleztem, de eddig nem vezette át, így valószínűleg nem is fogja.

Ahhoz, hogy a csomag legfrissebb változátával is használhassuk a módosítást a következőt kell tenni.

- Le kell tölteni a csomag forrás változatát. (`RODBC...tar.gz`)
- Kibontás után az `src/RODBC.c` fileban a következő változásokat kell átvezetni:

  A file elején a `__declspec(dllimport) window RConsole;` sort le kell cserélni és néhány sort kikommentezni. Csere után ez a rész a következőképpen néz ki:

```
#ifdef WIN32
# include <windows.h>
# undef ERROR
// comment out the next few lines in the original source and include
// the R_interactive line below.
/* enough of the internals of graphapp objects to allow us to find the
   handle of the RGui main window */
```

```
//typedef struct objinfo {
// int kind, refcount;
// HANDLE handle;
//} *window;
//__declspec(dllimport) window RConsole;
__declspec(dllimport) extern int R_Interactive;
#else
# include <unistd.h>
#endif
```

Az `RODBCDriverConnect` függvény elejét módosítani kell a kommentekkel megjelölt helyeken:

```
SEXP RODBCDriverConnect(SEXP connection, SEXP id,
                        SEXP useNRows, SEXP ReadOnly){
    SEXP ans;
    SQLSMALLINT tmp1;
    SQLRETURN retval;
    SQLCHAR buf1[buf1_len];
    pRODBCHandle thisHandle;

    PROTECT(ans = allocVector(INTSXP, 1));
    INTEGER(ans)[0] = -1;
    if(!isString(connection)) {
       warning(_("[RODBC] ERROR:invalid connection argument"));
       UNPROTECT(1);
       return ans;
    }
    thisHandle = Calloc(1, RODBCHandle);
    ++nChannels;

    odbcInit();
    retval = SQLAllocHandle(SQL_HANDLE_DBC, hEnv, &thisHandle->hDbc);
    if(retval == SQL_SUCCESS || retval == SQL_SUCCESS_WITH_INFO) {
      if(asLogical(ReadOnly))
        SQLSetConnectAttr(thisHandle->hDbc, SQL_ATTR_ACCESS_MODE,
          (SQLPOINTER) SQL_MODE_READ_ONLY, 0);
         //insert the next three lines
          //it gets the desktop window handle.
#ifdef WIN32
         HWND desktopHandle = GetDesktopWindow();
#endif
         retval =
           SQLDriverConnect(thisHandle->hDbc,
#ifdef WIN32
             // insert the next line and comment the original
             R_Interactive? desktopHandle : NULL,
             //RConsole ? RConsole->handle : NULL,
#else
             NULL,
```

```
#endif
            /* This loses the const, but although the
                declaration is not (const SQLCHAR *),
                it should be. */
             (SQLCHAR *) translateChar(STRING_ELT(connection, 0)),
             SQL_NTS,
             (SQLCHAR *) buf1,
             (SQLSMALLINT) buf1_len,
             &tmp1,
#ifdef WIN32
               // insert the next line and comment the original
               R_Interactive ? SQL_DRIVER_COMPLETE : SQL_DRIVER_NOPROMPT
#else
               SQL_DRIVER_NOPROMPT
#endif
             );
```

A rutin többi részét nem kell változtatni.

- Az így módosított file-t a kibontott csomag többi részével együtt vissza kell csomagolni, pl. a `RODBC.tar.gz` név alatt.
- Forrásból kell telepíteni a csomagot a következő paranccsal.

```
install.packages("RODBC.tar.gz", repos=NULL, type="source")
```

10.2. Bizonytalan kapcsolatfelvétel Rstúdióval, 2014-03-11

EXCEL-ből kezdeményezve az összekötést egy futó Rstúdió példánnyal előfordulhat, hogy a kapcsolat létrehozása sikertelen. Ez nálam nem jelentkezett, de az Rstudio 0.98.501 változatát telepítve én is megtapasztaltam. A hiba okát nem sikerült teljesen felderíteni, de annyi kiderült, hogy a `R.xls` munkafüzet `RIC` moduljában a `cmdToR` eljárásában kell keresni. A következő változat nálam gond nélkül működik. A változtatás a `.useSendKey` ágon belül `SendKeys "{ESC}", True` sorban történt. Korábban itt `"{ENTER}"` szerepelt.

```
Private Sub cmdToR(cmd)
  On Error GoTo Errorhandler
  With curRproc
    If .useSendkey Then
      Dim wstate
      wstate = WINAPI.WindowState(.hwnd)
      WINAPI.ShowWindow .hwnd, WINAPI.SW_NORMAL
      AppActivate curRproc.pid
      If VBA.Len(cmd) > 1 And .useClipboard Then
        Dim x As New DataObject
        SendKeys .sendkeyPrefix, True
        If .delay > 0 Then
          DoEvents
        End If
        x.Clear: x.SetText cmd: x.PutInClipboard
        SendKeys "{ESC}", True
```

```
        SendKeys .ctrlv, True
        SendKeys "{ENTER}"
        DoEvents
        WINAPI.ShowWindow .hwnd, wstate
      Else
        SendKeys .sendkeyPrefix
        SendKeys "{ENTER}"
        SendKeys quote(cmd)
        SendKeys "{ENTER}"
      End If
    Else
      WINAPI.SendText vbCrLf & cmd & vbCrLf, .hwnd
    End If
  End With
  Exit Sub
Errorhandler:
  Dim errnum, errtxt
  errnum = err.Number: errtxt = err.Description
  err.Clear
  Select Case errnum
    Case 91 ' obj does not exist
      If allRproc Is Nothing Then RprocInit: Resume
      If allRproc.Count = 0 Then Rstart: Resume
    Case 5 '??? mi a kodja annak amikor ay appactivate okozza a hibat
      With curRproc
        If (WINAPI.IsWindow(.hwnd) > 0) And _
           (WINAPI.IsWindowVisible(.hwnd) <= 0) Then
          WINAPI.ShowWindow .hwnd, WINAPI.SW_SHOWMINIMIZED
          Resume
        End If
      End With
  End Select
  MsgBox "Nem lehet írni az R konzolba!" & vbCrLf & _
         errtxt & "[errnum=" & errnum & "]", vbOKOnly
  End
End Sub
```

10.3. VBA módosítás, 2014-03-11

Az `R.xls` munkafüzet `RIC` modulja kapott egy szubrutint `showRprocs` névvel. Ez megjeleníti a futó `R` példányokat. Az `Rxls development` fül alá beraktam egy gombot, amivel ez a funkció kényelmesen elérhető. Ugyanebben a modulban az `RConnectedToThis` és `RnotBusy` függvénynek adtam egy opcionális `Rproc` típusú argumentumot, ha nem adjuk meg, akkor a `curRproc` példányt használják a rutinok, különben a megadottat. A `selectR` dialóg gombjait pedig felcseréltem, felülre raktam a `Cancel` gombot.

10.4. `VBA` módosítás, 2014-05-13

A `Calc.xlt` sablont atneveztem `Calc.xltm`-re így a számoló munkafüzetek ezentúl nem kompatibilis módban jönnek létre. Az `Rdev.xlam` bővítmény `insertSkeleton` moduljában a `newSkeleton`

rutinjából a mentés részt kivettem. Viszont a Calc.xltm beillesztettem egy Workbook_beforeSave esemény kezelőt. Így, amikor egy újonnan létrehozott számoló munkafüzetet először mentünk el, annak típusa makróbarát munkafüzet lesz. A SaveAs dialóg nem .xlsx, hanem a .xlsm kterjesztést kínálja fel alapértelmezésben.

10.5. R 3.1 változat LENGTH or similar ... hibaüzenet, 2014-09-01

A hibaüzenetet az okozta, hogy a com és comproxy csomagok C kódja az R API LENGTH függvényét használja több ponton. Az R 3.1-es változatában, újdonságként, ebbe a függvénybe belekerült egy ellenőrzés, ami error üzenetet ad, ha a függvényt nem vektor típusú SEXP-re alkalmazták. A LENGTH mellett az R API-ban van egy másik függvény is az Rf_length, amit a bővítmény csomagokban javasolnak alkalmazni.

A következő R szkripttel lehet megnézni a két függvény viselkedését. A kódrészlet végén az új, 3.1-es R alatt látszik a futási eredmény.

```
require(inline)
require(Rcpp)
funs<-c("LENGTH","Rf_length")
names(funs)<-paste("test",funs,sep="_")
body.pat<-'
  SEXP sexp = R_NilValue;
  sexp = R_MakeExternalPtr(NULL,R_NilValue,R_NilValue);
  return IntegerVector::create( _["%s"]=%s(sexp));
'
body<-lapply(funs,function(x)sprintf(body.pat,x,x))
test<-cxxfunction(sig=lapply(funs,"[",0), body=body, plugin="Rcpp")
names(test)<-funs
for(x in names(test)){
  cat(sprintf("Running 'test$%s':\n result: ",x))
  cat(eval(substitute(try(test[[x]](),silent=T),list(x=x))))
}
Running 'test$LENGTH':
 result: Error in test[["LENGTH"]]() :
  LENGTH or similar applied to externalptr object
Running 'test$Rf_length':
 result: 1
```

Az R korábbi változataiban az eredmény LENGTH esetén 0, míg Rf_length esetén 1 lett volna. Ez azért nem okozott gondot, mert ahol externalptr object előfordulhatott ott valójában azt nézte a program, hogy a hosszra valami teljesül-e, vagy externalptr-ről van-e szó.

A C, C++ forrásokban kicseréltem a LENGTH függvényhívásokat Rf_length-re. Ezzel a problémát sikerült orvosolni.

10.6. VBA módosítás, 2014-09-01

A kapcsolatfelvétel az Rstúdióval nem megnyugtató. Ha még semmi nem került az R consolba, akkor akár {ENTER}, akár {ESC} leütést küldünk, az Rstúdió leáll és új R sessiont kell indítani. Ezért kivettem a korábban jó megoldásnak tűnő „sendkey {ESC},True" sort az R.xls munkafüzet RIC moduljának cmdToR eljárásából, lásd a 10.2 szakaszt.

10.7. EXCEL módosítás, 2015-09-01

- Az Rselect űrlapban az R verzióját is feltüntetjük. Rstudio esetén csak akkor, ha a com csomag be van töltve.
- Javitás az Rproc modulban. Ha az Rstudio „projekt módban" dolgozik, akkor is megtaláljuk az ablak azonosítóját.
- A R.RIC modul connect rutinja módosult. Létrehozza az adott EXCEL példány környezetét az R oldalon. Ha menüből indítjuk, akkor meghívja az attachXL függvényt is. Ezzel az R keresési útvonalához csatolja a THISXL, .wbname, .wsname változókat.
- A R.RIC modul Rclose rutinja módosult. Törli az EXCEL példányra mutató környezeteket és az elérési útról is eltávolítja ezeket az adatokat. Ahol lehet a COM interfészt használja.
- A R.WINAPI modul sendkeystohwnd rutinja robosztusabb lett.

plusz további változtatások.

11 Mellékletek listája

A leírás, az R csomagok, a minta projekt elemei elérhetőek a következő címen

http://hpz400.cs.elte.hu/AEGON

Az oldal jelszót kér:

username: AEGON
password: R+EXCEL

```
A 'pkgs' könyvtár tartalma:
   com_1.0-80.tar.gz
   com_1.0-80.zip
   comproxy_1.0-26.tar.gz
   comproxy_1.0-26.zip
   jaradek2013_2.0-33.tar.gz
   jaradek2013_2.0-33.zip
   Rxls_2.0-292.tar.gz
   Rxls_2.0-292.zip
```

```
A 'Rfiles' könyvtár tartalma:
   calc1.R
   Calc1.xls
   ell.R
   kar.R
   lx2003.csv
   R_chunks_in_Rxls_user_guide.R
```

Az RStudio projekt honlapja és letöltési oldala:

http://www.rstudio.com/
http://www.rstudio.com/ide/download/desktop
http://www.rstudio.com/ide/download/preview

Az Rtools letöltési oldala:

http://CRAN.R-project.org/bin/windows/Rtools/

A Debugview letöltési oldala:

http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx