Vállalati gyakorlat I.

Elemző keretrendszer tervezése és implementálása

Készítette: Soltész Tamara

Témavezetők:

Bencsik Gergely
Pödör Zoltán

Tartalom

1. Feladat leírása

A féléves feladatom a Cyclic Reverse Moving Intervals Techniques (CReMiT) és a Random Correlations (RC) módszerek összekapcsolása és az összekapcsolás alapján interdiszciplináris elemzések tervezése, végrehajtása. A CReMIT módszer az alapadatok körének bővítésével kiterjeszti az elemzési lehetőségek körét, ami azonban maga után vonja, hogy a nagy mennyiségű adatsorok közötti kapcsolatok keresése során véletlen összefüggések is megjelenjenek. Utóbbiak felfedését és szűrését teszi lehetővé az RC módszer. A korábbiakban a két módszer egymástól függetlenül került fejlesztésre és alkalmazásra, az én feladatom a két módszer egy egységes adatkezelési keretrendszerbe történő integrálása és ennek egy egységes fejlesztői környezetben történő megvalósítása.

2. Felhasznált módszerek

2.1. CReMIT

Cyclic Reverse Moving Interval Techniques [1] rövidítése, mely a csúszó átlagot veszi alapul. Bemenete egy periodikus idősor, szükségszerűen időbélyeggel ellátva, valamint a felhasználó által bekért “maximális eltolás (I)”, “maximális ablakszélesség (J)”, és kezdőpont (SP) paraméterek, melyek egyértelműen meghatározzák a módszer aktuális futását.

CReMIT - pszeudo algoritmus:

```
CReMIT(Tömb(szám) adatsor, Egész SP,I,J,periodus) :
     segedhossz=(adatsor_hossza-SP)/periodus
     Egész index=0
     Tömb(szám) szarmaztatott= új Tömb hossza[I*J*segedhossz]
     ciklus i=0-tól I-ig :
          ciklus j=0-tól J-ig :
               ciklus k=SP-től adatsor_hossz-ig, periodus léptékkel:
                    Tömb(szám) tomb = új Tömb hossza[j+1]
                    ciklus d=0-tól tomb_hossz-ig :
                         tomb[d]=adatsor[k-d]
                    ciklus vége.
                    szarmaztatott[index]=Aggregalo(tomb)
                    index=index+1
               ciklus vége.
               Korreláció()
          ciklus vége.
     ciklus vége.
     vissza szarmaztatott
```

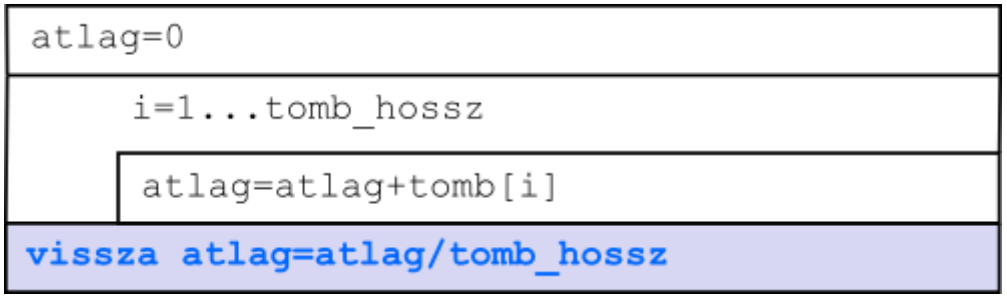
CReMIT algoritmus szöveges leírása (saját)

Ahol “adatsor” a beolvasott elemek tömb halmaza, “SP” a kezdő érték, “I” a maximális eltolás, “J” a maximális ablak szélesség, “periodus” a periódus hossza és “szarmaztatott” a származtatott adatsor halmaza. Az “Aggregalo()” és a “Korreláció()” függvények, amelyek leírása lentebb található.
A programomban ezt az algoritmust kicsit máshogy írtam meg, mivel míg itt a számolásokra, ott inkább a kiírásra fektettem nagyobb hangsúlyt, természetesen ettől még a számolás is fontos, és megfelelő.

Aggregálás
Az egyik függvény, amit a CReMIT algoritmus használ. Az adatok grafikonjának simítására szolgál a “csúsztatások” mellett. A programban 3 féle aggregáló függvény közül lehet választani, rádiógombok segítségével:

- ***Átlagolás***: a megadott tömb elemeinek átlagát adja vissza.
- ***Maximum*** *kiválasztás*: a megadott tömb elemei közül a legnagyobbat adja vissza.
- ***Minimum*** *kiválasztás*: a megadott tömb elemei közül a legkisebbet adja vissza.

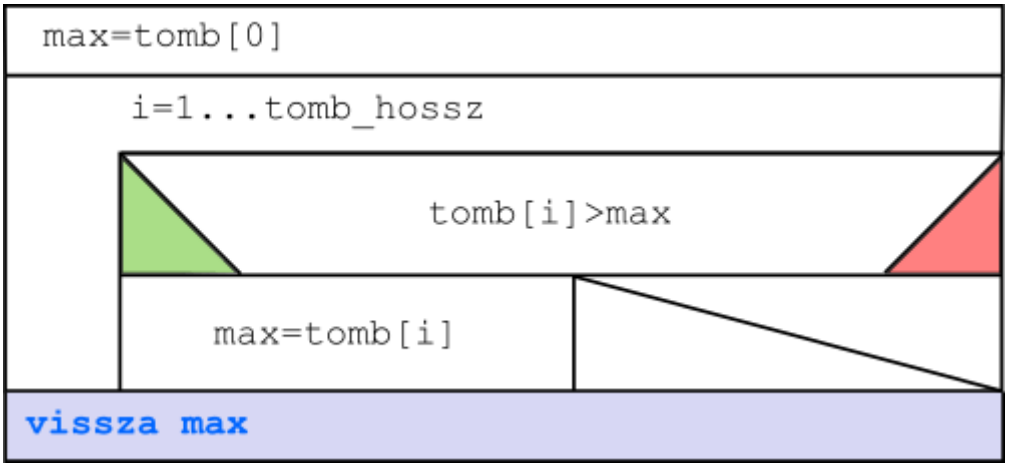
Átlagolás algoritmusa
Az “összegzés tételét” felhasználva:



1. ábra: Átlagolás struktogram (saját)

Ahol “tomb[]” egy tömb, amelynek elemeit átlagolni szeretnénk. Ezt a függvény meghívásakor kell paraméterként megadni.

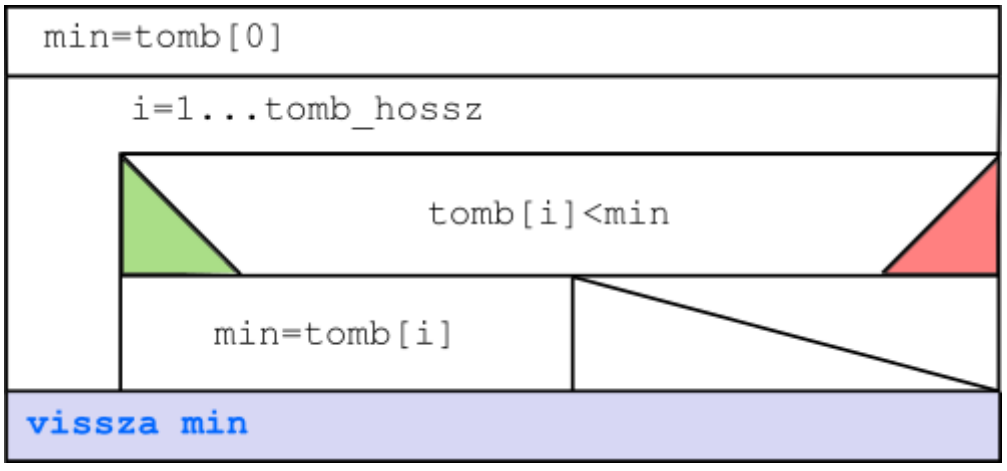
Maximum kiválasztás algoritmusa A “maximum kiválasztás tételét” felhasználva:



2. ábra: Maximum kiválasztás struktogram (saját)

Ahol “tomb[]” egy tömb, melyben szeretnénk megtalálni a legnagyobb elemet. Ezt a függvény meghívásakor kell paraméterként megadni.

Minimum kiválasztás algoritmusa
A “minimum kiválasztás tételét” felhasználva:



3. ábra: Minimum kiválasztás struktogram (saját)

Ahol “tomb[]” egy tömb, melyben szeretnénk megtalálni a legkisebb elemet. Ezt a függvény meghívásakor kell paraméterként megadni.
Ezeket mind egy függvényből lehet meghívni a programban és a rádiógombok állapotától függően hajtódnak végre. (Az “Aggregalo()” függvényt a programban “F()”-nek neveztem el).

```
762     public double F(double tomb[]){
763         if (jRadioButton1.isSelected()){
764             double atlag=0;
765                 for (int i=0;i<tomb.length;i++){
766                     atlag+=tomb[i];
767                 }
768             atlag/=tomb.length;
769             return atlag;
770         }
771         else if (jRadioButton2.isSelected()){
772             double max=tomb[0];
773             for (int i=1;i<tomb.length;i++){
774                 if (tomb[i]>max){
775                     max=tomb[i];
776                 }
777             }
778             return max;
779         }
780         else if (jRadioButton3.isSelected()){
781             double min=tomb[0];
782             for (int i=1;i<tomb.length;i++){
783                 if (tomb[i]<min){
784                     min=tomb[i];
785                 }
786             }
787             return min;
788         }
789         else {
790             return 0;
791         }
792     }
```

4. ábra: Az "`F()`" függvény programkódja (saját)

Korrelációszámítás

Ez ugyan nem szerves része a CReMIT módszernek, azonban kapcsolódik hozzá, mert az előállított, másodlagos adatsorokon ezzel a metodikával tudunk összefüggéseket keresni. A korreláció megadja két érték közötti lineáris kapcsolat nagyságát és irányát, illetve, hogy ez a két adat mennyire függ egymástól.

A korreláció a következő képlettel számítható ki:

$$R = \frac{\sum_{j=1}^{n} (a_j - \underline{a}) * (b_j - \underline{b})}{(n-1) * \sigma_a * \sigma_b}$$

$$\underline{\sigma_a} = \sqrt{\frac{\sum_{j=1}^{n} (a_j - \underline{a})^2}{n-1}}$$

$$\underline{\sigma_b} = \sqrt{\frac{\sum_{j=1}^{n} (b - \underline{b})^2}{n-1}}$$

$$\underline{a} = \frac{\sum_{j=1}^{n} a_j}{n}$$

$$\underline{b} = \frac{\sum_{j=1}^{n} b_j}{n}$$

Ahol n az adatok hossza. Jelen esetben nekünk R^2 -re van szükségünk.

```
1031        public double korelacio(double a[], double b[]){
1032            double osszega=0;
1033            double osszegb=0;
1034            for (int i=0;i<a.length;i++){
1035                osszega+=a[i];
1036                osszegb+=b[i];
1037            }
1038            double atlaga=osszega/a.length;
1039            double atlagab=osszegb/a.length;
1040            double sigma=0;
1041            double sigmb=0;
1042            for (int i=0;i<a.length;i++){
1043                sigma+=Math.pow(a[i]-atlaga,2);
1044                sigmb+=Math.pow(b[i]-atlagab,2);
1045            }
1046            sigma=Math.sqrt(sigma/(a.length-1));
1047            sigmb=Math.sqrt(sigmb/(b.length-1));
1048
                double kor=0;
1050            for(int i=0;i<a.length;i++){
1051                kor+=((a[i]-atlaga)*(b[i]-atlagab));
1052            }
1053            kor/=((a.length-1)*sigma*sigmb);
1054            return (kor*kor);
1055        }
1056
```

5. ábra: A "`Korrelacio()`" programkódja (saját)

2.2. RC módszer

A módszer [2] alkalmas arra, hogy az adatsorok közötti véletlen összefüggésekre felhívja a figyelmet. a véletlen összefüggések módszere. Érdekes tény, hogy az "összefüggések" aránya nagyobb, mint a "nem függ össze" legtöbbször.
Többféle módszer létezik erre.

Számítási megközelítések I.

1. A bemeneti adatsorból kiválasztjuk a legkisebb és legnagyobb elemet.
2. A közte lévő összes lehetséges variáció generálásra kerül "N" darabszámra.
3. A generált adatsorokra végrehajtjuk a megadott elemzési módszert.
4. Döntést kell hozni az összefüggésről.
5. Elosztjuk az "összefügg"-esek darabszámát az összes lehetséges eset darabszámával

Számítási megközelítések II.

1. Első adatsort beletesszük egy halmazba.
2. Következő adatsor ha össze függ az előző adatsorral akkor beleteszi az első halmazába, ha nem új halmazba tesszük.
3. Az új adatsor abba a halmazba kerül, amelyikben talál legalább egy olyan adatsort mellyel korrelál.
4. Ezt addig ismétli ameddig van adatsor.
5. A halmazok darabszáma +1 adatsor biztos össze függést eredményez legalább két adatsor között.

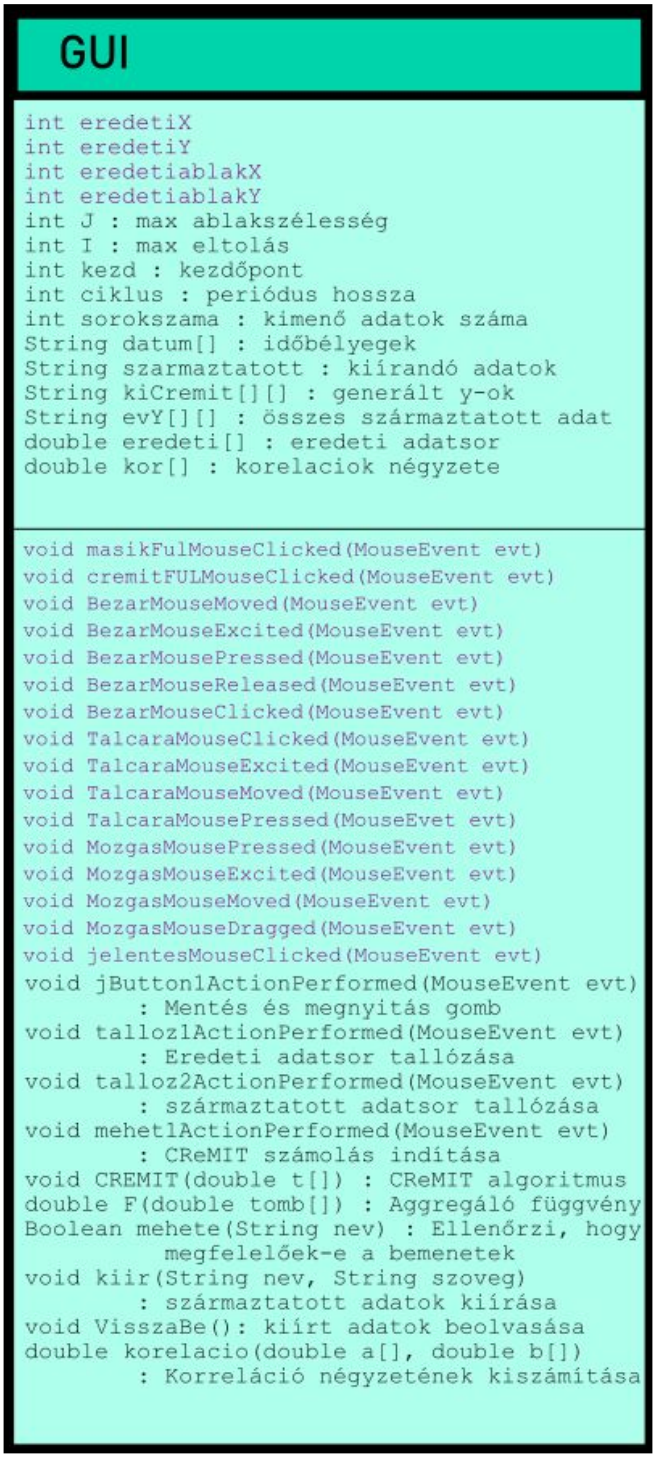
Számítási megközelítések III.

1. A bemeneti adatsorra elvégezzük a megadott vizsgálatot.
2. Elhagyjuk az utolsó elemet.
3. A maradt elemekre is elvégezzük a vizsgálatot.
4. Döntéshozás.
5. Ameddig van az adatsornak eleme az utóbbi három műveletet elvégezzük.
6. Ha a döntések gyakran változnak, akkor az adott módszerre az adatsor nem stabil.

3. A program felépítése

A program NetBeans IDE-ben készült, JAVA-t használva.
A munkám alapvető célja, hogy egy olyan egységes felületet hozzak létre, mely magában integrálja a CReMIT és az RC módszereket, azok minden lehetőségével együtt. A program két nagy részre osztható. Egyik az “agy”, melynek nagyobb része fent már említett és a másik pedig a grafikus felület.

Előbbihez még tartoznak függvények, amelyek összességét egy UML diagram ír le:



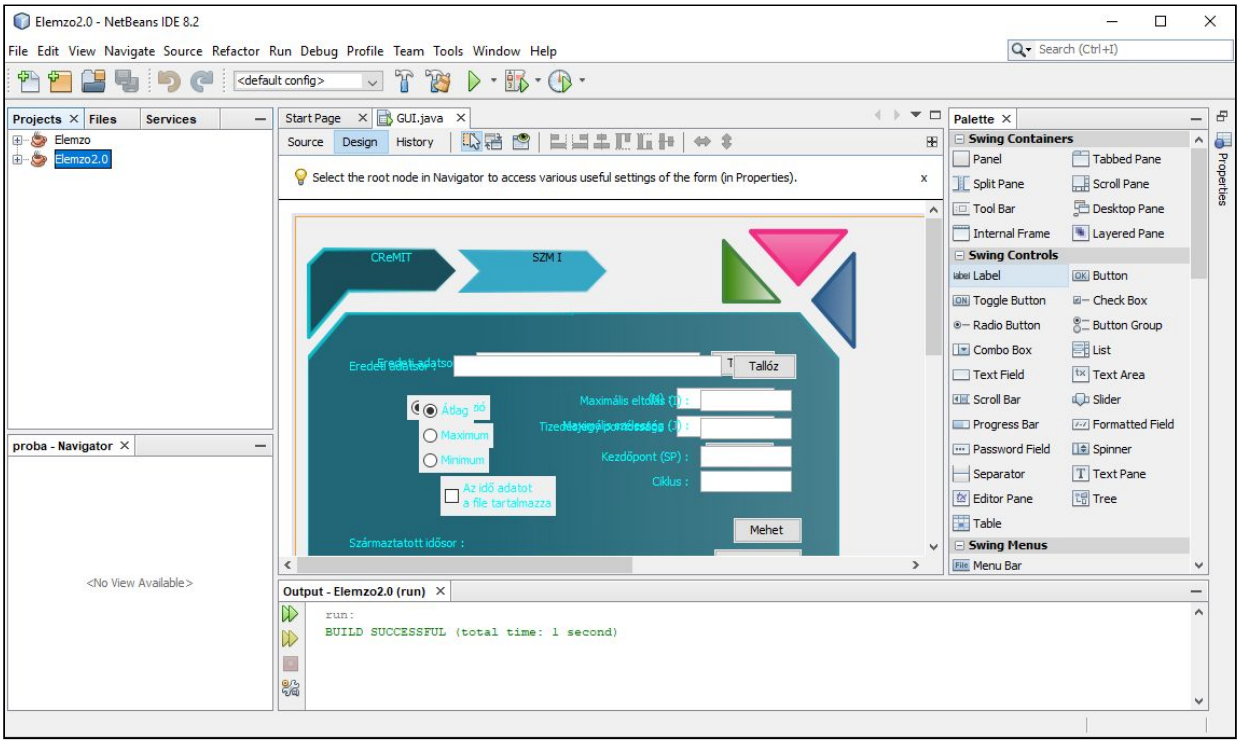
1. ábra: UML diagram (saját)

A program egy osztályból áll.

A függvények funkciói megtalálhatóak a diagramban, kivétel a lilával kijelöltek. Ezek “csak” grafikai szempontból fontosak, ugyanis a programablak bezárásáért, mozgatásáért és minimalizálásáért felelnek, valamint a program (mint lejjebb látható) több panelből áll, ezért a fülek megfelelő megjelenítésére is fel vannak készítve.

Grafikus felület (GUI):

A grafikus felület létrehozásához a JAVA JFrame osztálya kelt segítségre, melyet viszonylag könnyen lehet szerkeszteni a NetBeans “grafikus felület kezelő”-jében. Ehhez az osztályhoz tartoznak különféle J-komponensek, például gomb, rádiógomb, rajzfelület, íráshoz alkalmas felület és még sorolhatnám. A programban jelenesetben mindegyik felület megtalálható amik említve vannak, valamit sima szöveg felület. Az alkalmazást feldobva egyedi háttér készült hozzá az Inkscape nevezetű rajzoló programban. Minden egyes olyan rajzolt komponens külön készült mely feliratkozik egy eseményre (egész pontosan egy OnClick eseményre, mely mint a nevében is benne van kattintáskor aktiválódik).



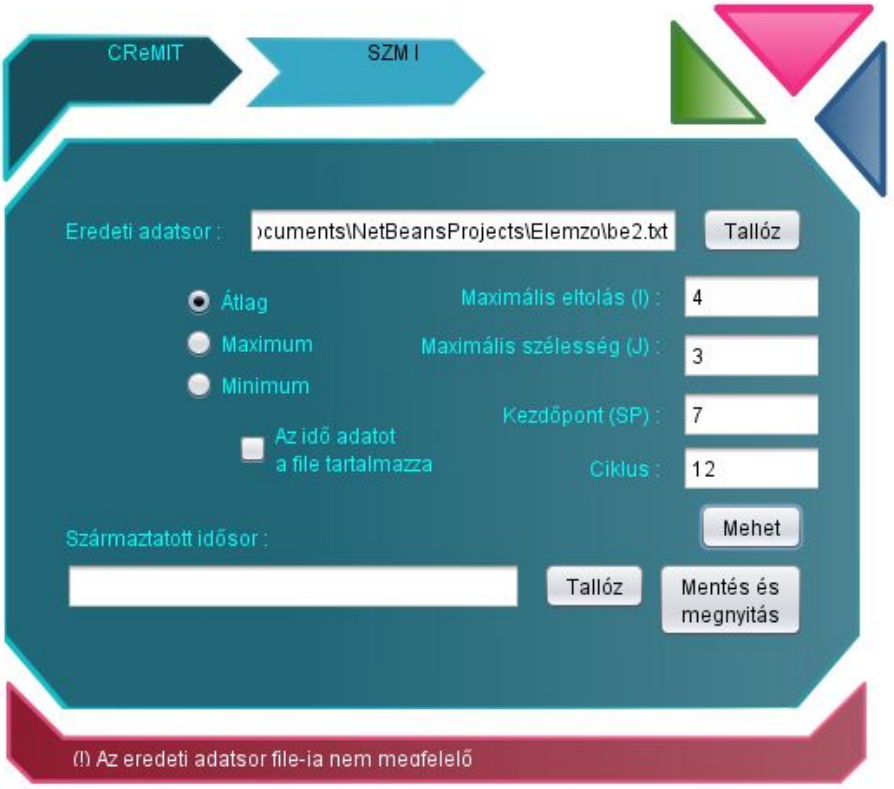
2. ábra: A NetBeans grafikus szerkesztőjének nézetéből

Egyéb különlegességéhez tartozik, hogy ez nem egy tipikus “négyzet ablak”, hanem a rajzolt elemek között nincsen a programhoz tartozó elem, így ha oda kattintunk úgy érzékeli mintha “kívülre kattintanánk” az ablakból. Ezt úgy érhetjük el, hogy a JFrame hátterét teljes egészében átlátszóvá tesszük a setBackground(new Color(0,0,0,0)); paranccsal. A rajzolt komponensek egy-egy JLabelen helyezkednek el. Ezek szinte mindegyikéhez külön-külön tartozik egy-egy OnClick esemény, melyek lentebb lesznek kifejtve felhasználói nézetből. Ezek mellett még említés tennék arról is, hogy nem csak maga a háttér átlátszó, de az alkalmazás látható részei is átmenetesen átláthatóak, de fontos tényező, hogy sötét és világos háttéren is jól mutat, nem megfeledkezve a mintás háttérről sem. Ezekhez egy-egy példát lent láthatunk majd.

A felhasználó nézetéből:
A program indítása után ez a kép látható

3. ábra: Az alkalmazás fekete háttéren

1. A tallózás gombra kattintva ki kell választani az adatfájlt. Ennek a fájlnak .txt kiterjesztésűnek kell lennie, valamint tabulátorral legyen elválasztva a két benne lévő adat egymástól, a következő adat párnak az utána lévő sorban kell lennie és figyeljünk oda, hogy az utolsó sor az utolsó adat pár legyen. Az első adat az idő , másik a számoláshoz használt érték legyen.
2. Ki kell választani, hogy melyik aggregáló függvénnyel szeretnénk dolgozni.
3. "Az idő adatot a file tartalmazza" checkboxot pipáljuk ki ha az adatsor rendelkezik időbélyeggel.
4. Írjuk be a "maximális eltolást", "maximális szélességet", "kezdőpontot" és a "ciklust". Mindegyiknek pozitív egész számnak kell lennie.
5. Ha mindent rendben találtunk, kattintsunk a "mehet" gombra. Ha valamit hibásan adtunk meg, (akár nem pozitív egész számot, vagy nem megfelelő fájl elérési útvonalat és még sorolhatnám) alul a hiba jelentő sávban jelzi nekünk és kiírja azt. Ha el szeretnénk tüntetni a hibát, csak kattintsunk a jelentő sávra és visszakapjuk az eredeti színét, hibaüzenet nélkül.



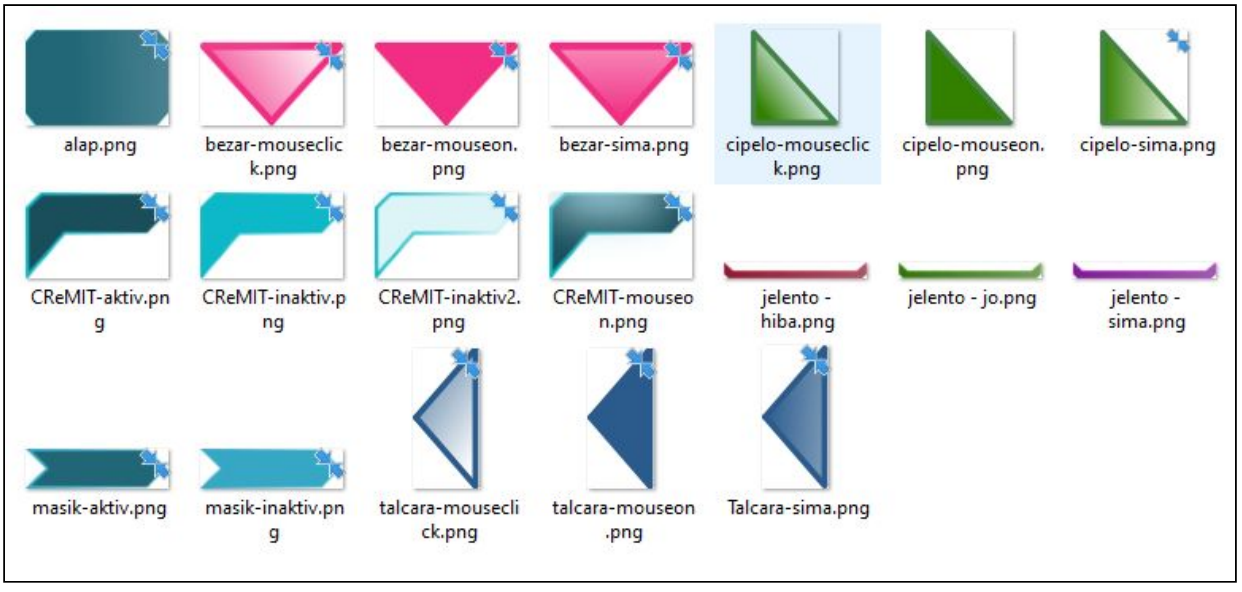
4. ábra: Az alkalmazás hibás input esetén (fehér háttéren)

1. Ha sikerült akkor a jelentő sáv zöldre vált és a “Kész” felirat jelenik meg rajta és mehetünk tovább. (A “Kész” üzenetet is eltüntethetjük, úgy ahogy az 5. pontban le van írva)
2. Az alsó “Tallóz” gombra kattintva kiválaszthatjuk hova mentsük az eredményeinket.
3. “Mentés és megnyitás” gombra kattintva elmenti az adott helyre a fájlt, majd megnyitja azt szövegszerkesztőben. (Hiba esetén szintén jelzi a jelentő sávban)

5. ábra: Az alkalmazás sikeres mentés után (mintás háttéren)

Egyéb funkciók:

- Az ablakot a zöld háromszög húzásával lehet pozícionálni.
- Az ablakot a ciklámen háromszögre kattintással lehet bezárni.
- Az ablakot a kék háromszögre kattintással lehet a tálcára tenni.
- Ha az egerünket bármely háromszög felé vezetjük, a színe megváltozik. Valamint ha megnyomjuk az egerünket, szintén más árnyalat látható, tehát összesen 3 féle állapota lehet a gombnak (“inaktív”,”készenléti”, “aktív”).
- A panelek változtatását a fülekre történő kattintással lehet elérni. Az aktív panel füle mindig a sötétebb szín.



6. ábra: A felhasznált grafikai elemek

4. További fejlesztési lehetőségek

A program jelenlegi állapotában a CReMIT módszert valósítottam meg a Pearson-féle korrelációs összefüggés kereséssel, azonban még ezt is fejleszteni kell/lehet az alábbi elemekkel:

- Ne csak az időbélyeggel ellátott adatsorokat fogadja el, hanem lehessen vele generáltatni megfelelő idő intervallumon belül.
- Felismerje automatikusan, hogy a bemeneti fájlban melyik az időbélyeg, melyik az adat.
- Két külön kimeneti fájlja legyen és ezeket külön lehessen átnevezni, vagy akár választani melyik legyen elmentve.
- Bővíteni kell az aggregációs függvények körét, legalább az összeg függvénnyel.

Meg kell valósítani az RC komponenst, ügyelve arra, hogy a CReMIT módszerhez kötődő korrelációs elemzés kimenete legyen ennek a bemenete.

Maga a program felület is fejleszthető, például az alábbi elemekkel:

- Akár excel fájlból olvasni, vagy kiíratni közvetlenül.
- A program második felének elkészítése.
- Nem beépített gombok használata, hanem egyedi dizájnt rajzolni azoknak is.

5. Irodalomjegyzék

[1] PÖDÖR, Z. – EDELÉNYI, M. – JEREB, L. Systematic analysis of time series – CReMIT. In *Infocommunication Journal*. Vol. 6 Iss. 1, March 2014, p. 16-21.

[2] BENCSIK, G. – Bacsárdi, L. Novel methods for analyzing random effects on ANOVA and regression techniques. In *Advances in Intelligent Systems and Computing*. Vol. 416, January 2016, p. 499-509.