LAPATINSZKI ANDRÁS

Interaktív szófelhő alkalmazás készítése R-ben szövegbányászati feladatokhoz

Tartalomjegyzék

1. Előszó 4

2. Bevezető 5

2.1. Szövegbányászat 6

2.1.1. Szövegbányászat folyamata 7

2.1.2. A szófelhő 9

2.1.3. Dokumentum-kifejezés mátrix 10

2.2. Használt eszközök 10

2.2.1. R programozási nyelv 11

2.2.2. Shiny 11

2.2.3. RStudio 12

2.2.4. Roxygen 13

3. Használt csomagok 13

3.1. Csomag a szövegbányászathoz: tm 14

3.1.1. Korpusz tisztítása 14

3.1.2. Adathalmaz feldolgozása 14

3.1.3. Kifejezés-dokumentum mátrix 15

3.1. Webes alkalmazás keretrendszer R-hez: shiny 15

3.2. Dashboard készítése Shiny-ban: shinydashboard 15

3.3. Adatvizualizáció a „Grammar of Graphics” alapján: ggplot2 15

3.4. Egymást nem fedő szövegek és címkék a ’ggplot2’-nél: ggrepel 16

3.5. Hálózat analízis és vizualizáció: igparh 16

3.6. Számos R eszköz az adat vizualizációhoz: gplots 16

3.7. Függvények memória kezelésé: memoise 17

3.8. Letöltés az NCBI adatbázisból: RISmed 17

4. global.R 18

4.1. Szükséges csomagok telepítése, betöltése 18

4.2. Felhasználói felület beállítása 19

4.2.1. skin 20

4.2.2. header 20
4.2.3. sidebar 20
4.2.4. body 23
4.3. Adathalmaz feldolgozás 26
4.3.1. Kifejezés-dokumentum mátrix készítése 26
4.3.2. Rendezett mátrix a kifejezés-dokumentum mátrixból 27
4.3.3. Data Frame készítése 27
4.3.4. Szomszédsági hálózat 29
4.4. A szófelhő elkészítése 30
4.5. Beolvasott fájl kiterjesztésének meghatározása 31
4.6. Interakciós részek 31
4.6.1. 1.5 sugarú környezet 31
4.6.2. A koordinátához tartozó szó 32
4.6.3. Adott szóhoz tartozó előfordulási szám 33
5. server.R 34
5.1. NCBI korpuszműveletek 35
5.1.1. Nagyításhoz használt segédváltozók 35
5.1.2. Rendezett kifejezés-dokumentum mátrix az NCBI adatbázisból 35
5.1.3. Data Frame az NCBI adatbázisból 36
5.2. NCBI szófelhő 36
5.3. NCBI interakciók 37
5.3.1. Az NCBI szófelhő nagyítása 37
5.3.2. Az NCBI szófelhőhöz tartozó kattintási információk átadása 37
5.4. NCBI asszociációs hálózat 38
5.4.1. Asszociációs hálózat az NCBI adatbázisból 38
5.4.2. Asszociációs hálózat átadása az NCBI-ból 39
5.5. Fájl kiválasztása, beolvasása 40
5.6. Kiválasztott fájlhoz tartozó korpuszműveletek 41
5.6.1. Data Frame készítése a kiválasztott fájlból 42
5.7. Szófelhő a kiválasztott fájlból 42

5.7.1. Szófelhő készítése a kiválasztott fájlból 42
5.7.2. Szófelhő átadása a kiválasztott fájlból 43
5.8. Interakciók a szófelhőhöz 43
5.8.1. A kiválasztott fájlból készült szófelhő nagyítása 43
5.8.2. A kiválasztott fájlhoz tartozó kattintási információk átadása 44
5.9. A kiválasztott fájl asszociációs hálózata 44
5.9.1. Asszociációs hálózat készítése a kiválasztott fájlból 44
5.9.2. Asszociációs hálózat átadása kiválasztott fájlból 46
5.10. Mentés pdf vagy png fájlként 47
6. Összefoglalás 48
6.1. Szövegbányász adatstruktúrák 48
6.2. Adattisztítás 48
6.3. Interaktív szófelhő és szó korreláció 48
6.4. Medline adatbázis 49
6.5. GUI: Grafikus felhasználói felület 49
6.6. A program használata 49
6.7. További lépések 55
7. Irodalomjegyzék 56
8. Függelék/Mellékletek 58
9. Nyilatkozat 59

1. Előszó

Feladatomnak egy interaktív szófelhő alkalmazás készítését vállaltam, ami szövegbányászati feladatokhoz használható. Az alkalmazás R nyelven készült, az úgynevezett RStudio fejlesztési környezetben.

Számtalan szófelhőalkalmazás létezik már, melyek hiába különböznek egymástól valamilyen szinten, lényegében mégis mind ugyan azt a feladatot végzik el egy adott szövegből: szógyakoriság alapú szófelhőt rajzolnak ki randomizált, vagy irányított elhelyezéssel. Másik közös tulajdonságuk az interaktivitás hiánya.

Hiába jeleníti meg minden egyes szófelhő az általa feldogozott szöveg leggyakoribb szavait. A szavak összessége hiába rajzol ki egy szabályos kört, egy madarat, vagy egy szívet, ha utána a megjelenített címkékből semmi más információ nem nyerhető ki.

Választott témám célja, hogy az alkalmazás, az ábrázolt szófelhőből olyan adatokat adjon vissza a felhasználó felé egyetlen kattintással, mint például maga a megjelenített szó, a hozzá tartozó gyakoriság és az adott szóhoz tartozó asszociációs hálózat.

Az alkalmazáshoz még hozzá tartozik, hogy különböző kiterjesztésű állományok feldolgozására legyen képes. Különböző kiterjesztésű okmányokat tudjon beolvasni és utána feldolgozni, mint például: txt, csv, pdf és html. Az ábrázolás után pedig legyen lehetséges a szófelhő nagyítása, valamint a kirajzolt felhő menthető legyen pdf és/vagy png állományként.

Szeretnék köszönetet mondani Kisander Zsoltnak és Dr. Feldmann Ádámnak a türelmükért, az általuk nyújtott segítségért és támogatásért. Mind az R nyelv és az RStudio megismerésében, a szövegbányászat témakörbe való bevezetésben és hozzá tartozó források ajánlásában, az alkalmazás írása közbeni folyamatos tanácsokban és iránymutatásokban.

2. Bevezető

Már számtalan megvalósított szófelhő létezik. Még az R-en belül is akadnak szófelhő előállítására alkalmas csomagok, mint a ’wordcloud’, ’wordcloud2’. Ezen csomagok az R szövegbányászati csomagokat használják a korpusz[1] tisztítására, a szavak előfordulási számának kinyerésére.

A ’wordcloud” képes egy esztétikus, jól rendezett szófelhő megjelenítésére, de teljesen hiányzik belőle az interakció. A vizualizáció egyetlen képként jelenik meg. Nem tudjuk megállapítani a pontos előfordulási számot, nem tudunk kifejezéseket kijelölni benne.

A ’wordcloud2’ mondható a ’wordcloud’ fejlesztett verziójának. Egy nagyon szép és személyre szabható szófelhő készíthető vele. Egyedi maszkok készíthetőek hozzá, amik megadják a szófelhő alakját. Itt nem csak egy körre, vagy négyzetre kell gondolni. Készíthető vele akár madár, Eiffel torony, csillag, fa formájú szófelhő. Itt már vannak kezdeti lépések az interakció felé. Egy adott kifejezés fölé helyezve a kurzort, egy ablakban megjelenik a kifejezés, és a hozzá tartozó előfordulási szám. Viszont ezek kinyerésére nincs lehetőség, a csomagban található egyik függvénynek sincs ezen értékekre vonatkozó visszatérése.

Az általam készített szófelhőben az interakción van a hangsúly. Természetes módon, a későbbiekben szeretnék fejleszteni a vele való vizualizáció esztétikáján, de a feladat szempontjából nem ez a leglényegesebb teendő. Az alkalmazás így nagymértékben könnyíteni tudja a fontos információ kinyerését a nagy terjedelmű anyagoból. Elősegítheti az Medline adatbázisban való keresést. Használatával csökkenthető a számunkra felesleges információt tartalmazó publikációk, dokumentációk, tanulmányok kiszűrésére fordított idő.

[1] Korpusz: Olyan dokumentumok gyűjteménye, melyek a szövegbányászati folyamatául szolgálnak.

2.1. Szövegbányászat

„A szövegbányászat a strukturálatlan vagy kis mértékben strukturált szöveges állományokból történő ismeret kinyerésének tudománya. Olyan különböző dokumentumforrásokból származó szöveges ismeretek és információk gépi intelligenciával történő kigyűjtése és reprezentációja, amely a feldolgozás előtt rejtve és feltártalanul maradt az elemző előtt.” (Wikipedia, Szövegbányászat, dátum nélk.)

Mivel az elektronikus formában tárolt adatok egyre növekvő hányadát a szöveges dokumentumok teszik ki (e-mail, emlékeztető, üzleti és kutatási beszámoló, prezentáció, hírek, reklámanyag, weboldal stb.), ezért egyre nagyobb igény van olyan megoldásokra, amelyekkel hatékonyan lehet szövegeket intelligens módon feldolgozni és elemezni.

„Az egyszerű keresésnél jóval többet hivatott nyújtani a szövegbányászat. Míg szöveges keresés esetében meglévő információkra kívánunk kis időbefektetéssel rátalálni (nagy relevanciájú találati eredmények által), addig a szövegbányászat során olyan tudásra, ismeretekre is szert kívánunk tenni, ami explicit módon nem volt benne a rendelkezésre álló dokumentumállományban (korpuszban), csak indirekt reprezentációként, rejtve, látensen. Bár a teljes szövegű keresés is a szövegbányászat része, a szövegbányászat a keresésnél jóval többet jelent, hasonlóan, ahogy az adatbányászat többet jelent az egyszerű adatkeresésnél.” (Wikipedia, Szövegbányászat, dátum nélk.)

„A szövegbányászat nagymértékben épít az adatbányászat eredményeire, ahol elsősorban számszerű adatok feldolgozása történik gépi tanulási módszerekkel. Az adatbányászat azon eredményeit, amelyek minták felismerésére, adatreprezentációra, predikcióra, statisztikai összefüggések kimutatására vonatkoznak, a szövegbányászat is nagymértékben hasznosítja. A különbség abban mutatkozik, hogy míg adatbányászat esetében jól strukturált számszerű adatokkal dolgozunk, addig a szövegbányászatban strukturálatlan szöveges állományok képezik a kiindulási alapot.” (Wikipedia, Szövegbányászat, dátum nélk.)

Ma már szinte minden fontosabb statisztikai szoftver tartalmaz szövegbányász lehetőségeket. A legfontosabb ilyen lehetőségek:

- előzetes feldolgozás (preprocess): az adatok általános előkészítése, importálása, tisztítása;
- társítások, kapcsolódások (associate): kapcsolódási analízis, ami egy adott meghatározáshoz keres jellemző kapcsolatokat, elsősorban gyakorisági és együttes előfordulási alapon;
- csoportosítás (cluster): a hasonló dokumentumok csoportosítása;
- összefoglalás (summarize): a legfontosabb fogalmak megkeresése (ezek tipikusan magasabb gyakoriságú fogalmak);
- kategorizálás (categorize): a szövegek előre meghatározott kategóriákba való besorolása;
- API lehetőségek (application programming interface): a programok bővítésének lehetősége plug-in-ek segítségével.

Az R/tm ezen lehetőségek mindegyikét tartalmazza.

(BDE Research Közhasznú Nonprofit Kft., Adatgyűjtés, elemzés, alaptechnológiák elemzése, 2010)

2.1.1. A szövegbányászat folyamata

Adatgyűjtés → Előkészítés → Elemzés → Megjelenítés

Adatgyűjtés

Az adatgyűjtésnél megkülönböztetünk webes és nem webről származó adatforrásokat.

- Nem webes adatforrások:
  - Ide sorolhatók a kérdőíves megkérdezések, ahol akár több ezer egymástól független, egy-két mondatos szöveges rekordot kell feldolgozni.
  - Olyan kutatási projektek, melyek a szövegbányászat lehetőségeit aknázzák ki, mint a naplóprojektek[2].

[2] Naplóprojekt: „Egy előre felállított minta állít elő bizonyos, kötött tematikájú, de akár hosszabb szövegeket.”

- Cégek által kutatási céllal rendelkezésre bocsátott szöveges dokumentumok: sajtótartalmak, levelezések, írásos kommunikációs elemek.

- Webről származó adatforrások:
  - Nem felhasználói tartalmak. Ez alatt a vállalati tartalmak, weboldalak tartalma és ágazatspecifikus tartalmak értendők.
  - Felhasználói tartalmak, melyekbe a tematikus blogok, a fórumhozzászólások és közösségi oldalak tartozna.
  - Webes keresési eredmények.

Előkészítés

A szövegbányászat kulcsfontosságú területe, hiszen a nyelvi jellegű gépi elemzés hatékonyságát nagyban befolyásolja az alkalmazott módszer és technológia.

Itt a nehézségek közé tartozik a különböző formátumú, eltérő hosszúságú szövegek, valamint a kifejezések különböző írásmódjainak kezelése, beleértve az elégpelést, a szinonimákat és a különböző szóalakokat.

A bementek kezelésére először az alapvető rendszereket, fő struktúrákat kell felállítani.

- Az első ilyen szint, a dokumentum gyűjtemény, más néven korpusz. A szövegtest mellett, a meta-adatokat (létrehozás dátuma, szerző, címe, a szöveg nyelve…) is magába foglalja.
- A következő szint a szöveges dokumentum. Itt a szövegek egy „text repository”-ba kerülnek. Célja az eredeti dokumentum tarolása, a visszakereshetőség biztosítására.
- Végül a kifejezés-dokumentum mátrix létrehozása. Itt előfeldolgozást végzünk a korpusz egyedi szavainak kinyerésére. Az előfeldolgozás lehet:
  - A szóközök és a kívánatos karakterek elkülönítése a szavaktól.
  - A korpusz szavainak redukálása a stopszavak[3] eltávolításával és a szótövezés[4] révén.

[3] Stopszavak: Az elemzés szempontjából nem releváns szavak, mint a névelők, névmások, kötőszavak, kérdőszavak. Vagyis az olyan gyakran előforduló szavak, melyek nehezítik az információkinyerés folyamtát.

[4] Szótövezés: A szavak szótőre való redukálása, elő- és utótagjaitól való megfosztása.

- Automatikus kulcsszógyűjtés: a korpuszt legjobban jellemző szavak azonosítása.
- A szavak súlyozása.
- A szinonimák kezelése.

Elemzés

Az elemzési technikák:

- Számosság alapú feldolgozás
  - A nagy gyakoriságú kifejezéseknek, meghatározásoknak tulajdonít nagyobb fontosságot.
- Információ kivonatolás
  - Kulcsfogalmak és azok viszonyainak azonosítása.
- Kategorizálás
  - A dokumentumok felügyeletlen, valamilyen hasonlósági mérték alapján történő csoportokba sorolása.
- Osztályozás
  - Többnyire felügyelt tanulási algoritmuson keresztül működik. Előre definiált csoportokba való besorolás. Például spam-szűrés.

Megjelenítés

Az elemzési technikát megfelelően reprezentáló megjelenítés választása.

(BDE Research Közhasznú Nonprofit Kft., Adatgyűjtés, elemzés, alaptechnológiák elemzése, 2010)

2.1.2. A szófelhő

A szófelhő egy vizualizációs eljárás, ami összegzi a megadott forrás/források kifejezéseit. Jellemzően weboldalakhoz tartozó kulcsszavak (címkék) ábrázolására vagy szabad formátumú szövegek megjelenítésére használják. Ezen címkék többnyire egyetlen kifejezést jelentenek. Csak a leggyakrabban előforduló N darab címkét jeleníti meg. Minden egyes címke jelentősége, gyakorisága az adott szövegben, a címke méretével és/vagy a színével, esetleg pozícionálásával fejezhető ki.

Ez a formátum segít a szövegben leggyakrabban előforduló kifejezések gyors megtalálásában és a relatív gyakoriságok meghatározásában.

(Trattner, Helic, & Strohmaier)

2.1.3. Dokumentum-kifejezés mátrix

A dokumentum-kifejezés mátrix vagy kifejezés-dokumentum mátrix egy matematikai mátrix, mely magába foglalja a megadott dokumentumokban található kifejezések előfordulási számát. A dokumentum-kifejezés mátrix sorai az egyes dokumentumoknak, míg az oszlopai az egyes kifejezéseknek felelnek meg (gyakran használjuk a mátrix transzponáltját is). Így jellemezhetővé válik a kifejezés fontossága. Számos módszer létezik, hogy kinyerjük az egyes kifejezések fontosságát. Az egyik ilyen módszer a TF-IDF[5] algoritmus.

Például:

Van két dokumentumunk, melyekből szeretnék elől állítani a dokumentum-kifejezés mátrixot.

A két dokumentumunk tartalma a következő:

- D1: „Ez az első példaszöveg.”
- D2: „Ez a második példaszöveg.”

A két dokumentumból, a következő dokumentum-kifejezés mátrixot kapjuk:

	Ez	az	a	első	második	példaszöveg
D1	1	1	0	1	0	1
D2	1	0	1	0	1	1

1. táblázat: *Dokumentum-kifejezés mátrix*

(Wikipedia, Document-term matrix, dátum nélk.)

2.2. Használt eszközök

Azért az R-nyelvet és az RStudio-t választottam, mert jól dokumentált. Számos elkészített csomag létezik hozzá, melyek segítségével gyorsabban, könnyebben lehet dolgozni. Aktív fejlesztés alatt áll, rendszeresen

[5] TF-IDF (**T**erm **F**requency–**I**nverse **D**ocument **F**requency): „Ez az érték jellemzi egy szó fontosságát az adott dokumentumban. A fontosság növekszik a szó újabb és újabb előfordulásával, de csökken, ha az adott korpuszon belül egyre több dokumentumban jelenik meg.” (BDE Research Közhasznú Nonprofit Kft., Algoritmusok leírása, 2011)

jelenik meg újabb verziója. A felhasználható csomagok száma folyamatosan gyarapszik. A szövegbányászati területen nagymértékben elterjedt, hiszen a 'tm' csomagban szereplő függvények teljes mértékben fedik a szövegbányászat szükséges lépéseit. Mind ezek mellett ingyenes, nyílt forrású és könnyen integrálható más funkciójú csomagokkal. Használata viszonylag egyszerű és könnyen megtanulható.

2.2.1. R programozási nyelv

Az R egy szakterület-specifikus programozási nyelv és szoftverkörnyezet a statisztikai számításokhoz és ábrázolásokhoz. Statisztikusok és az adatbányászok körében használatos adatelemzésekhez és statisztikai szoftverek fejlesztéséhez.

A R forráskódja elsősorban C, Fortan és R nyelveken íródott, ami szabadon hozzáférhető a GNU GPL-ben (General Public Licence) biztosított jogok szerint. Széleskörűen bővíthető különböző csomagok használatával. Az alapvető csomagokat az R telepítője tartalmazza, az ezeken felül létező csomagok pedig megtalálhatóak a CRAN-on, az „átfogó R archívum hálózaton".

Az R első változatát Ross Ihanka és Robert Gentleman készítette 1993-ban. Az utolsó stabil verzió (3.3.2) kiadása pedig 2016.10.31. Jelenleg több mint 8000 hivatalos statisztikai programcsomag érhető el rajta.

(Wikipedia, R (programozási nyelv), dátum nélk.)

2.2.2. Shiny

A Shiny egy nyílt forráskódú R csomag, ami egy elegáns és stabil webes keretrendszert biztosít a webes R alkalmazások készítéséhez és fejlesztéséhez.

A Shiny előnyei:

- Használatához nincs szükség JavaScript, HTML, CSS ismeretekre.
- Segítségével olyan interaktív webes alkalmazások készíthetőek, melyek képesek a dinamikus adatkezelésre.
- Előre felépített modulokkal rendelkezik a grafikus megjelenítéshez.
- Bármilyen R környezetben használható.
- A Shiny alkalmazások hostolhatóak helyileg, ön-hostolt Shiny szerveren, vagy RStudio-hostolt Shiny szerveren.

Egy Shiny alkalmazás felépítése:

Az alkalmazás két fő összetevője a felhasználói kezelőfelület meghatározása és a szerver utasítássor. Ezeken felül még tartalmazhat bármi plusz adatot, utasítássort vagy különböző szükséges kiegészítőket, amik segítik az alkalmazás futását.

- Felhasználói kezelőfelület (ui.R): Ebben az utasítássorban határozzuk meg a felhasználói kezelőfelület weblapon megjelenő kinézetét.
- Szerver (server.R): Ebben az utasítássorban a statisztikai modellezést és a kimenetek ábrázolását definiáljuk. A felhasználói felületről érkező adatokat dolgozza fel, majd az eredményt válaszként továbbítja a felhasználói felületre. Alapvető feladata, hogy meghatározza a bemenetek és kimentek közötti kapcsolatot
- Plusz utasítások kezelése (global.R): Tetszőleges, feladatspecifikus kódokat tartalmaz.

(Gorakala, 2014)

2.2.3. RStudio

Az RStudio egy ingyenes és nyílt forráskódú integrált fejlesztési környezet (IDE) az R programozási nyelvhez. C++ programozási nyelven íródott és a Qt[6] keretrendszert használja a grafikus felhasználói felületéhez. Tartalmaz konzolt és „snytax-highlighting" szerkesztőt, ami támogatja a közvetlen kódfuttatást, valamint ábrázolást, előzménykezelést és debuggolást. Létezik kereskedelmi verziója is.

Alapítója Joseph J. Allaire. Az első nyílvános béta verzió (v0.92) 2011 februárjában jelent meg. Az eddigi utolsó verziója pedig 2016.11.06-án.

Két kiadása létezik:

- RStudio Deskop: Aminél a program helyileg fut, mint bármilyen asztali alkalmazás.

[6]Qt („cute"): Egy cross-platform alkalmazás keretrendszer, amit leginkább olyan szoftverek fejlesztéséhez használnak, amik különféle szoftver és hardver platformokon úgy futtathatók, hogy csak kis mértékben, vagy egyáltalán nem változnak a mögöttes kód alapjaik. Így megtartva az eredeti alkalmazást az eredeti képességeivel és sebességével. (Wikipedia, Qt (software), dátum nélk.)

- RStudio Server: Ami web böngészőn keresztül engedélyezi az RStudio-hoz való hozzáférést, amíg az egy távoli Linux szerveren fut.

(Wikipedia, RStudio, dátum nélk.)

(RStudio, dátum nélk.)

2.2.4. Roxygen

Egy a Doxygen-hez hasonló dokumentáció generátor R környezetben. Használatával átlátható dokumentációs fájl és 'NAMESPACE' fájl készíthető a megfelelő formátumú kommentek írásával. A forráskód írása közbeni speciális kommentek alapján, automatikusan generálja le a dokumentációs fájlt, így nekünk nem kell külön dokumentációt készíteni.

3. Használt csomagok

Az alkalmazás elkészítéséhez több, már létező csomagot használtam fel, a gyorsabb haladás érdekében. Ezen csomagok a következők:

- tm – csomag a szövegbányászathoz;
- shiny – webes alkalmazás keretrendesz az R-hez;
- shinydashboard – dashboard készítése Shiny-ban;
- ggplot2 – adatvizualizáció a „Grammar of Graphics" alapján;
- ggrepel – egymást nem fedő szövegek és címkék a 'ggplot2'-nél;
- igrahp – hálózatanalízis és vizualizáció;
- gplot – számos R eszköz az adat vizualizációhoz;
- memoise – függvények memória kezelése;
- RISmed – letöltés az NCBI adatbázisból.

3.1. Csomag a szövegbányászathoz: tm

A tm csomag, különböző hasznos funkciókkal segíti a szöveges dokumentumok kezelését. Tartalmazza a szövegbányászati folyamatokhoz használatos dokumentumkezelést és lehetővé teszi a különböző formátumú szövegfájlok felhasználhatóságát az R nyelven belül.

A csomag integrált adatbázis háttértámogatással is rendelkezik, hogy minimalizálja a memória igényét. Egy fejlett meta-adat kezelő van implementálva benne, hogy a szöveges dokumentum előállításánál könnyítse a nagyméretű, valamint a sok meta-adattal rendelkező dokumentum csoportok kezelhetőségét.

3.1.1. Korpusz tisztítása

- content_transformer: Tartalom átalakítók létrehozása. Olyan függvények, melyek módosítják egy R objektum tartalmát.
- removeNumbers: Eltávolítja a szövegből a számokat.
- removePunctutation: Eltávolítja a szövegből az írásjeleket.
- removeSparseTerms: Eltávolítja a kifejezés-dokumentum mátrixban szereplő ritka kifejezéseket.
- removeWords: Eltávolítja a szövegből a megadott szavakat.
- stopwords: Változó stopszavakkal tér vissza. Több különböző nyelvet is támogat (magyar változata is elérhető). Alatta a névelőket, névmásokat, stb. értjük.
- tm_map: Módosító függvények alkalmazása a korpuszon.

3.1.2. Adathalmaz feldolgozása

- Corpus: Korpusz létrehozása.
- readPDF: Segítségével egy külső PDF olvasó program hívható meg, amely képes kinyerni a PDF-ből a szöveget és metadatot.
- VCorpus: Volatile Corpus, egy olyan korpusz létrehozása, melyen lényegesen sok módosítást szeretnénk eszközölni.

3.1.3. Kifejezés-dokumentum mátrix

- TermDocumentMatrix: Egy kifejezés-dokumentum mátrixot készít. Ami megadja, hogy az egyes kifejezések hányszor szerepelnek egy adott dokumentumban.

(Feinerer, 2015)

3.2. Webes alkalmazás keretrendszer R-hez: shiny

A csomag segítségével egyszerű utasításokkal építhető fel egy interaktív webes alkalmazás R nyelven. Automatikus „reaktív" kapcsolat készíthető a bemenetek és kimentek között. Valamint az alapos, előre felépített elemek lehetővé teszik egy szép, érzékeny, és erőteljes alkalmazás készítését minimális energiaráfordítással is.

(Chang, Package 'shiny', 2016)

3.3. Dashboard készítése Shiny-ban: shinydashboard

Úgynevezett dashboard[7] készítése Shiny-val. A csomag segítségével a Shiny felületre egy módosítható témát húzhatunk, egy előnyös, kellemesen kialakítású dashboard formájában.

(Chang, Package 'shinydashboard', 2016)

3.4. Adatvizualizáció a „Grammar of Graphics" alapján: ggplot2

A ggplot2 egy ábrázoló rendszer az R nyelvhez, aminek az alapja: *The Grammar of Graphics*. A csomagot igyekeztek úgy megalkotni, hogy az magába foglalja mind az alap, mind pedig a rácsos/hálós grafikus ábrázolás előnyeit, azok hátrányai nélkül. A csomag gondoskodik az olyan nehézkes részletek

[7] dashboard: „Kulcs információk egyszerű megjelenítése, ami segít a felhasználónak gyorsan, helyes döntéseket hoznia." (horvimi, 2013)

kezelésről, amelyek bonyolítják az ábrázolást (mint például a feliratok, jelmagyarázatok), valamint egy olyan erős grafikai modellel rendelkezik, ami lehetővé teszi az összetett, többréteges ábrázolást.

(Wickham, 2016)

3.5. Egymást nem fedő szövegek és címkék a 'ggplot2'-nél: ggrepel

Feliratokat és címkéket biztosít a 'ggplot2' csomagnak, hogy az elkerülje az egymást elfedő címkék ábrázolását. A címkék el vannak távolítva egymás közeléből, és az ábrázolt koordinátáktól is.

(Slowikowski, 2016)

3.6. Hálózat analízis és vizualizáció: igparh

Az igraph egy könyvtárgyűjtemény gráfok készítéséhez és kezeléséhez, valamint hálózat analízishez. C nyelven íródott és létezik Python illetve R csomagja is. Leginkább tudományos kutatásokban használatos. Csárdi Gábor és Nepusz Tamás fejlesztették.

Alapvető tulajdonságai:

- képes a nagy hálózatok hatékony kezelésére
- eredményesen használható magas szintű programozási nyelveken
- interaktív és nem interaktív használatot is támogat

(Csardi, 2015)

3.7. Számos R eszköz az adat vizualizációhoz: gplots

Számos, az R programozáshoz használatos eszközt tartalmaz az adat vizuális megjelenítéséhez.

A gplots csomagban megtalálhatóak:

- Függvénysimítás ('bandplot', 'wapply')
- Alap ábrázoló függvények fejlesztett verziói ('barplot2', 'boxplot2', 'heatmap.2', 'smartlegend'),
- Színek kezelése ('col2hex', 'colorpanel', 'redgreen', 'greenred', 'bluered', 'redblue', 'rich.colors'),
- Két-dimenziós adat összegzések számolása és ábrázolása ('ci2d', 'hist2d'),

- Fejlesztett regressziós elemző ábrázolás ('lmplot2', 'residplot'),
- Egyszerűsített elérési illesztés a ’stats::lowess’ függvényhez ('lowess'),
- Szöveges adatok megjelenítése ('textplot', 'sinkplot'),
- Olyan mátrixok ábrázolása, ahol mindegy egyes cella tartalmaz egy pontot, amely nagysága jelöli az adott elem relatív fontosságát ('balloonplot'),
- Venn diagrammok ábrázolása ('venn'),
- Open-Office stílusú ábrázolás ('ooplot'),
- Többszörös adat ábrázolása ugyanazon területen, elkülönített tengelyekkel ('overplot'),
- Ábrázoló eszközök és megbízhatósági intervallumok ('plotCI', 'plotmeans'),
- X-Y tengelyen ábrázolt pontok közti távolság létrehozása, hogy a pontok ne fedjék egymást ('space').

(Warnes, 2016)

3.8. Függvények memória kezelése: memoise

Optimalizáló eljárás a programok futási sebességének növelésére. A módszer lényege, hogy elraktározza a nagy időigényű függvények visszatérési értékét, és megegyező bementek esetén gyorsabban térjen vissza a megfelelő értékkel.

Egy változóba elmenti a megadott értéket. Ez a memorizált másolat lényegében az eredeti funkció egy hanyagabb verziója: ha a függvényt új értékkel hívják meg, elmenti a visszatérési értéket. A következő hívásra pedig kikeresi a megfelelő bemenetre adott régebbi választ. Nagy időigényű számításoknál növeli a lefutás sebességét.

(Hester, 2016)

3.9. Letöltés az NCBI adatbázisból: RISmed

Egy eszközkészlet bibliográfiai tartamlom kinyerésére az NCBI (National Center for Biotechnology Information) adatbázisokból, a PubMed-et is beleértve. A RISmed név a RIS (for Research Information

Systems, ami egy általános kifejezésforma a bibliográfiai adatokra) és a PubMed[8] szavak összevonásából jött létre.

(Kovalchik, 2016)

4. global.R

A megírt függvényeket tárolja. A server.R és az ui.R folyamatai innen hívják meg a szükséges függvényket.

4.1. Szükséges csomagok telepítése, betöltése

```
if(require(tm) == FALSE){
  install.packages("tm")
}
library(tm)
```

Ha a `require` függvény visszatérési értéke `FALSE`, akkor nem tudja betölteni a benne megadott csomagot, vagyis az nincs telepítve. Ebben az esetben telepítenünk kell azt.

Ha a `requrie` függvény visszatérési értéke `TRUE`, akkor a csomag már telepítve van, ezért ezt a lépést kihagyjuk.

Az ellenőrzés végén pedig betöltjük a megfelelő csomagot.

[8] PubMed: Egy szabad kereső motor, ami első sorbán MEDLINE adatbázis hivatkozásaihoz valamint tudományos és biológiai témáinak kivonataihoz fér hozzá. (Wikipedia, PubMed, dátum nélk.)

4.2. Felhasználói felület beállítása

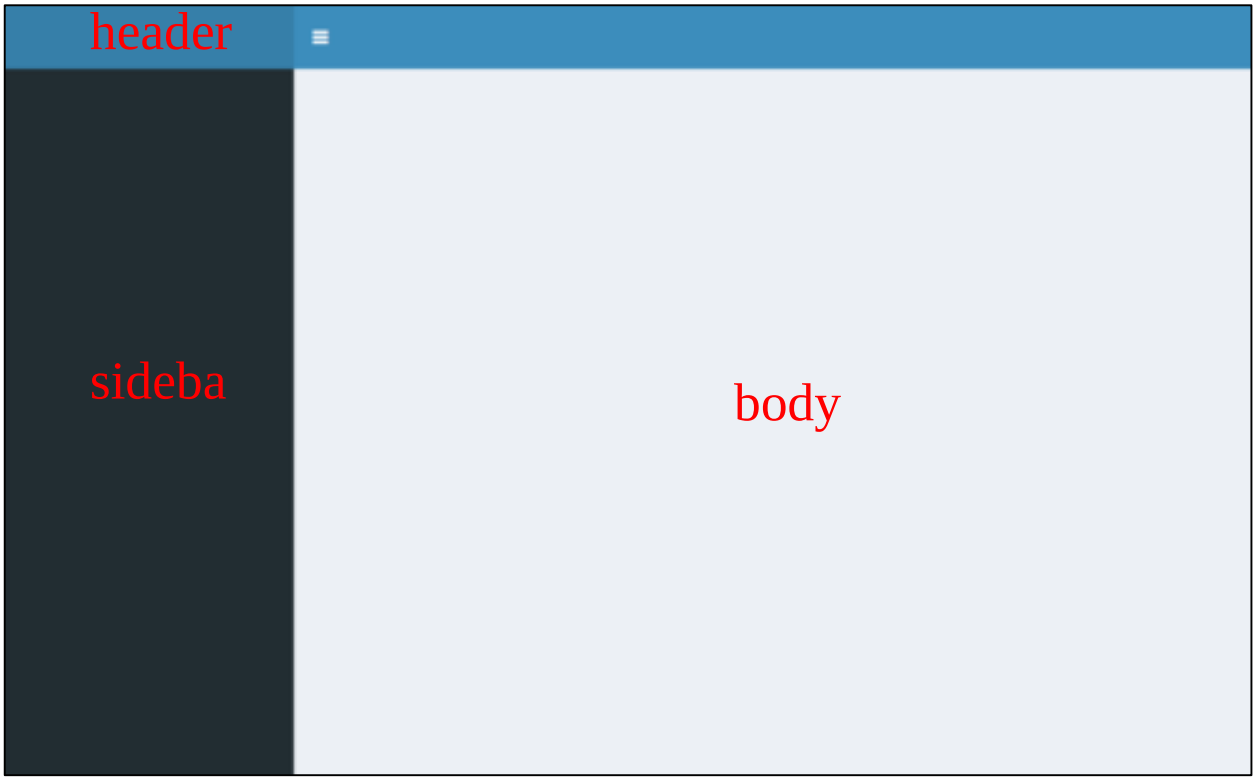
```
header <- dashboardHeader(...)

sidebar <- dashboardSidebar(sidebarMenu(...))

body <- dashboardBody(tabItems(...))
```

A Shiny Dashboard elkészítéséhez szükség van négy paraméterre.

- `header:` A dashboard fejléce.
- `sidebar:` Az oldalsáv.
- `body:` A dashboard fő része.



1. ábra: *Shiny dashboard szerkezete*

- `skin:` Színbeállítás, egyszerűen csak ki kell választani az előre beállított színekből egyet. Ezt az ui.R-ben tettem meg.

4.2.1. skin

```
fluidPage(
  dashboardPage(skin = "green", header, sidebar, body)
)
```

Mivel a grafikus felhasználói felület minden beállítása a golbal.R fájlban van beállítva (a `skin` -t kivéve), az ui.R-ben pedig csak a dashboardPage függvény van meghívva, ami a `skin` paraméterrel kiegészítve megkapja ezeket a beállításokat, ezért úgy tartottam logikusnak, ha itt említem meg.

A kiválasztható színek a következők:

2. ábra: *Dashboard színek*

4.2.2. header

```
header <- dashboardHeader(title = "MedLine Miner")
```

A fejlécen a „MedLine Miner” felirat beállítása.

4.2.3. sidebar

A menü elemeket egyesével részletezem, mert van olyan, ami csak egy sor, viszont van olyan is, ami hosszabb.

```
menuItem(strong("Wordcloud from File"), tabName = "fromFile"),
```

„Wordcloud from File” feliratú menü, félkövér betűstílussal. Ide kattintva, megváltozik a `body` felülete. A `tabName = "fromFile"` segítségével kapcsolódik össze a megjelenítendő elemekkel. Így az általunk kiválasztott fájlból készített szófelhő és asszociációs háló lesz látható.

```
menuItem("File Input",
         fileInput("infile", label=("")),
         actionButton("fileupdate", "Update")),
```

Egy „File Input” feliratú, legördülő menü. Segítségével választhatjuk ki az elemezni kívánt fájlt, valamint tartalmaz egy gombot, melyre kattintva erősítjük meg a kiválasztást.

```
  menuItem(strong("From NCBI"), tabName = "fromNCBI"),
```

„From NCBI” feliratú menü, félkövér betűstílussal. Ide kattintva, megváltozik a `body` felülete. A `tabName = "fromNCBI"` segítségével kapcsolódik össze a megjelenítendő elemekkel. Így az NCBI adatbázis választott forrásából készült szófelhő és asszociációs háló lesz látható.

```
  menuItem("NCBI adatbázis",
           selectInput("selection", label=(""), choices = db)),
```

„NCBI adatbázis” feliratú, legördülő menü. Ide kattintva választhatjuk ki a feldolgozás forrását.

```
  menuItem("Options",
       numericInput("num", label = h4("Maximális találat:"),
                    value = 100),
```

„Options” feliratú, legördülő menü. Ide kattintva állíthatjuk be az NCBI adatbázisból való keresés paramétereit.

Létrehozunk egy szám beviteli mezőt, ahol megadhatjuk a maximális találti számot, melynek alapértelmezett esetben 100 az értéke.

```
              dateInput('date',
                        label = paste('Keresés kezdete: ',
                                      'dd/mm/yy formátumban'),
                        value = as.character(Sys.Date()),
                        min = Sys.Date() - 5, max = Sys.Date() + 5,
                        format = "dd/mm/yy",
                        startview = 'year', language = 'en-US',
                        weekstart = 1
                        ),
              textInput("text", label = h4("Kulcsszavak"),
                        value = ""),
              actionButton("update", "Feldolgoz")
  ),
```

Ez után egy idő beviteli mezőt készítünk. Ennek segítségével állíthatjuk be, hogy mikori dokumentumokat jelenítsem meg a keresés. Alapértelmezett esetben a beírt dátum a rendszeridőből kinyert dátum lesz. Beállítjuk a minimálisan és maximálisan bevihető dátumokat. A formátumot beállítjuk

„napok/hónapok/évek”-re úgy, hogy mindegyikből két karakter jelenjen meg. Így például a 2017. január 7. a következő képpen jelenik meg: 07/01/17. Megjelenítjük az egész éves nézetet ’en-US’ nyelvre beállítva. Végül megadjuk, hogy a hét kezdő napja a hétfő.

```
            textInput("text", label = h4("Kulcsszavak"),
                      value = ""),
            actionButton("update", "Feldolgoz")
  ),
```

Ebben a menüben az utolsó beviteli mezőbe írhatjuk be a keresendő kifejezést. Alapértelemezett esetben ez a mező üres. Alatta pedig egy „Feldolgoz” feliratú gombot helyezünk, mellyel megerősíthetjük a bevitt paramétereket és megkezdhetjük a keresést.

```
  menuItem("Assoc options",
           textInput("assoc1", label = h4("Szóasszociációk"),
                     value = ""),
                numericInput("asnum1",label = h4("Level:(0-1)"),
                             value = 0.7),
                actionButton("booleButton","Network")
  ),
```

„Assoc options” feliratú, legördülő menü. Ide kattintva állíthatjuk be az asszociációs hálózat paramétereit:

- Egy szöveges bemeneten az asszociációs hálózat kiindulási szavát.
- Az asszociációs hálózathoz tartozó korrelációs határt.

Tartalmaz egy „Network” feliratú gombot, melyre kattintva erősítjük meg a beállított értékeket.

```
  menuItem("Sliders",
           sliderInput("freq",
                       "Minimum Frequency:",
                       min = 1,  max = 50, value = 20),
           sliderInput("max",
                       "Maximum Number of Words:",
                       min = 1,  max = 300,  value = 40),
           sliderInput("corlim",
                       "Correlation limit:",
                       min = 0, max = 1, value = 0.8)
  ),
```

„Sliders” feliratú, legördülő menü. Ide kattintva állíthatjuk be (csúszkák segítségével):

- Milyen minimális előfordulási értékkel kerüljenek be a kifejezések a szófelhőbe.
- A szófelhőbe kerülő szavak maximális számát.
- A szófelhőbe kerülő szavak korrelációs határát.

```
menuItem("Save",
         radioButtons("radio", label = (""),
                      choices = list(".pdf", ".png")),
         downloadButton("down","Save")
),
```

„Save” feliratú, legördülő menü. Ide kattintva választhatjuk ki, hogy az aktuálisan kirajzolt szófelhőt és asszociációs hálózatot milyen formátumba szeretnénk menteni. Pdf és png formátumok közül választhatunk. A „Save” feliratú gombra kattintva erősíthetjük meg a kiválasztott formátumot.

4.2.4. body

Kiválasztott fájlból

Itt három „box”-ot jelenítek meg egy oldalon a következő képpen:

```
tabItem(tabName = "fromFile",
        fluidRow(
          box(title = "Wordcloud", status = "success",
              solidHeader = TRUE,
              plotOutput("plot5",
                         click = "plot5_click",
                         dblclick = "plot5_dblclick",
                         hover = "plot5_hover",
                         brush = brushOpts(
                           id = "plot5_brush",
                           resetOnNew = TRUE
                         )
              )
          ),
```

Az első címét „Wordcloud”-ra állítom. Beállítom a színét (`status = "success"`), majd kap egy keretet. Itt jelenik meg a kiválasztott fájlból készült szófelhő, melyen engedélyezve van, hogy visszaadja a kattintás koordinátáit, a dupla kattintás koordinátáit, az egér pozíciójának koordinátáit a kirajzolt terület

felett, valamint az egérrel kijelölt területet. A szélességét nem kellett beállítani, hiszen az alapértelmezett értke az ablak fele (6; a teljes ablakszélesség 12).

```
            box(title = "Assoc", status = "warning",
                solidHeader = TRUE,
                svgPanZoomOutput("graph")
            ),
```

A második címe „Assoc”. Beállítom a kívánt színt (`status = "warning"`), majd kap egy keretet. Az asszociációs hálózatot SVG-ként jelenítem meg, hiszen itt nem volt szükség az egérinformációk visszaszerzésére. Valamint így nagymértékben nagyítható a görgő segítségével.

```
            box(h4("Info"), width = 12,
                status = "info", solidHeader = TRUE,
                verbatimTextOutput("info5"))
        )
),
```

A harmadik az „Info” címet kapta. A szélességét beállítom egész ablakosra. Mivel az előző két „box” már kitöltötte egy sorban a teljes ablakszélességet, ezért ez az előző kettő alá kerül.

Beállítom a színét (`status = "info"`), majd kap egy keretet. Itt jelennek meg az egérkattintással kinyer információk a szófelhőből (maga a szó és a hozzá tartozó előfordulás).

NCBI adatbázisból

Itt három „box”-ot jelenítek meg egy oldalon a következő képpen:

```
tabItem(tabName = "fromNCBI",
        fluidRow(
          box(title = "Wordcloud", status = "success",
              solidHeader = TRUE,
              plotOutput("plot",
                         click = "plot_click",
                         dblclick = "plot_dblclick",
                         hover = "plot_hover",
                         brush = brushOpts(
                           id = "plot_brush",
                           resetOnNew = TRUE
                         )
              )
          ),
```

Az első címét „Wordcloud”-ra állítom. Beállítom a színét (`status = "success"`), majd kap egy keretet. Itt jelenik meg az NCBI adatbázisból készült szófelhő, melyen engedélyezve van, hogy visszaadja a kattintás koordinátáit, a dupla kattintás koordinátáit, az egér pozíciójának koordinátáit a kirajzolt terület felett, valamint az egérrel kijelölt területet. A szélességét nem kellett beállítani, hiszen az alapértelmezett értke az ablak fele (6; a teljes ablakszélesség 12).

```
            box(title = "Newtork plot", status = "warning",
                solidHeader = TRUE,
                svgPanZoomOutput("plot3")
            ),
            box(h4("Info"), width = 12, status = "info",
                solidHeader = TRUE,
                    verbatimTextOutput("info"))
        )
  )
```

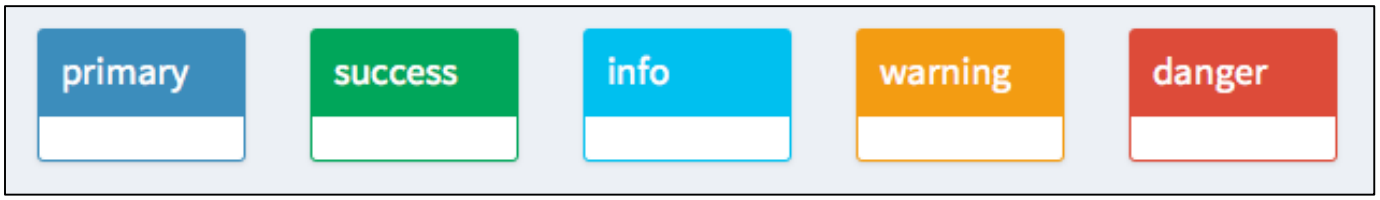
A második címe „Assoc”. Beállítom a kívánt színt (`status = "warning"`), majd kap egy keretet. Az asszociációs hálózatot SVG-ként jelenítem meg, hiszen itt nem volt szükség az egérinformációk visszaszerzésére. Valamint így nagymértékben nagyítható a görgő segítségével.

```
            box(h4("Info"), width = 12, status = "info",
                solidHeader = TRUE,
                    verbatimTextOutput("info"))
        )
  )
```

A harmadik az „Info” címet kapta. A szélességét beállítom egész ablakosra. Mivel az előző két „box” már kitöltötte egy sorban a teljes ablakszélességet, ezért ez az előző kettő alá kerül.

Beállítom a színét (`status = "info"`), majd kap egy keretet. Itt jelennek meg az egérkattintással kinyer információk a szófelhőből (maga a szó és a hozzá tartozó előfordulás).

A `status`-szal beállítható színek a következők:



3. ábra: *„Status” színek*

4.3. Adathalmaz feldolgozás

Ide tartozik a korpusz kinyerése, annak tisztítása, rendezése. Az így kapott kifejezés-dokumentum mátrix Data Farme-é alakítása, valamint a szomszédsági hálózat elkészítése.

4.3.1. Kifejezés-dokumentum mátrix készítése

```
myTermDocumentMatrix <- function(myData,frompdf = 0,uri = NULL){
  if(frompdf == 1){
    myCorpus <- VCorpus(URISource(uri, mode = ""),
            readerControl = list(reader = readPDF(engine = "xpdf")))
  } else {
    myCorpus = Corpus(VectorSource(myData))
  }

  myCorpus = tm_map(myCorpus, content_transformer(tolower))
  myCorpus = tm_map(myCorpus, removePunctuation)
  myCorpus = tm_map(myCorpus, removeNumbers)
  myCorpus = tm_map(myCorpus, removeWords,
                    c(stopwords("SMART"), "this", "these",
                      "that", "here", "and", "but"))

  myTDM = TermDocumentMatrix(myCorpus,
                             control = list(minWordLength = 2))
  return (myTDM)
}
```

A függvény három paraméterrel rendelkezik:

- `myData`: Az adathalmazt adja meg, melyből ki szeretnénk nyerni a mátrixot.
- `frompdf`: Segédváltozó, 0 alapértelmezett értékkel. Ha nélküle hívjuk meg a függvényt, ezt az értéked adja át neki. A változó megadja, hogy pdfből szeretnénk-e elkészíteni a mátrixot.
- `uri` (Uniform Resource Identifier): Segédváltozó, NULL alapértelmezett értékkel. Ha nélküle hívjuk meg a függvényt, ezt az értéket adja át neki. Pdf esetén, a külső pdf olvasónak (xpdf[9]) szükség van a dokumentum elérési útvonalra, amit ebben a változóban tárolunk.

[9] xpdf: A pdfinfo és pdftotext alkalmazások telepítése szükséges a használatához. A tm csomag alapesetben nem tartalmazza.

Ha pdf fájlból szeretnénk előállítani a mátrixot, akkor azt a `VCorpus` segítésével tudjuk megtenni. Meg kell adnunk hozzá az elérési útvonalat, valamint a külső pdf olvasót, ami egy txt kiterjesztésű fájlt hoz létre, hogy a feldolgozás elvégezhető legyen. Ellenkező esetekben, a `Corpus` parancs segítségével hozzuk létre a kívánt adathalmazból a korpuszt.

Ezek után elvégezzük a megfelelő átalakításokat:

- A szövegben található nagybetűket kicseréljük kisbetűkre.
- Eltávolítok az összes írásjelet.
- Eltávolítjuk a számokat.
- Eltávolítjuk az angol stopszavakat. A kötőszavakat, mutató névmásokat, és a többi önmagában lényeges jelentéssel nem bíró kifejezést.

Végül létrehozzuk a kifejezés-dokumentum mátrixot a megmaradt szavakból, melyek hossza legalább két karakter.

4.3.2. Rendezett mátrix a kifejezés-dokumentum mátrixból

```
getTermMatrix <- memoise(function(adat){
  m = as.matrix(adat)
  sort(rowSums(m), decreasing = TRUE)
})
```

A függvény paramétere:

- `adat`: A kifejezés-dokumentum mátrix, amiből a rendezett mátrixot szeretnénk készíteni.

A `memoise` segítségével érhetjük el, hogy ugyanazon bemenő értékekre ne fusson le újból a függvény, hanem csak keresse elő az adott bemenethez tartozó, előzetesen lementett visszatérési értéket.

Az `as.matrix` paranccsal, a kívánt kifejezés-dokumentum mátrixot egy mátrix objektummá alakítjuk, amit a `sort` paranccsal, előfordulás szerint csökkenő sorrendbe állítunk.

4.3.3. Data Frame készítése

Egy táblázatot hoz létre a szöveg szavaiból, amely tartalmazza, hogy az adott szó hányszor fordult elő a szövegben.

```
getMyDataFrame <- function(data,maxword,minfreq){
  myFrame <- as.data.frame(data)
  myFrame <- tibble::rownames_to_column(myFrame, "VALUE")
  colnames(myFrame) <- c("word","freq")
  myFrame <- subset(myFrame, freq>=minfreq)
  rownames(myFrame) <- myFrame$word
  N <- nrow(myFrame)-maxword
  if(nrow(myFrame)>maxword){
    myFrame <- head(myFrame, -N)
  }
  Xcord <- sample(1:maxword)
  Ycord <- sample(1:maxword)
  myFrame <- cbind(myFrame, X=c(Xcord), Y=c(Ycord))
  myFrame[1,3] <- maxword/2
  myFrame[1,4] <- maxword/2
  return (myFrame)
}
```

A függvény három paraméterrel rendelkezik:

- `data`: Az adathalmaz, amiből a táblázatot szeretnénk elkészíteni.
- `maxword`: Megadja, hogy maximálisan hány szóból álljon a táblázat.
- `minfreq`: Megadja, hogy minimálisan hányszor kell előfordulnia az adott szónak a szövegben, hogy a táblázatba kerülhessen.

Először az `as.data.frame` paranccsal elkészítjük a teljes szövegből a táblázatot. Ahhoz, hogy később elérhetőek legyenek a szavak, ezért azokat bemásoljuk a táblázat első oszlopába. Így a második oszlopba kerül az előfordulásuk száma. A táblázat első oszlopát átnevezzük „word”-re, míg a másodikat „freq”-re.

Ezek után formázzuk a táblázat terjedelmét. A `subset` paranccsal kivesszük azokat a szavakat, melyek előfordulása kisebb, mint a `minfreq` értéke. A táblázat sorainak csökkentéséhez meg kell néznünk, mekkora az aktuális táblázat. Ha a táblázatunk több sort tartalmaz, mint a `maxword` értéke, akkor az utolsó N darab sort kivesszük.

A táblázat elemeinek grafikus ábrázolásához X és Y koordinátákat rendelünk. A `sample` parancs, 1-től a `maxword` értékéig véletlenszerűen kiosztja a koordináta párokat. Miután végigért, a legelső szóhoz (ami a leggyakrabban fordul elő) hozzárendeljük a később kirajzolandó terület közepének a koordinátáit.

Például:

	word	freq	X	Y
data	data	45	6	6
frame	frame	37	1	4
bemutatása	bemutatása	29	3	2
egy	egy	26	4	5
egyszerű	egyszerű	23	5	1
példán	példán	23	2	3

2. táblázat: *Data Frame szerkezete*

4.3.4. Szomszédsági hálózat

```
getNetwork <- memoise(function(adat,corlimit,maxword){

  myTDM2 = removeSparseTerms(adat, sparse = corlimit)
  matrixTD <- as.matrix(myTDM2)
  N <- nrow(matrixTD)-maxword
  if(nrow(matrixTD)>maxword)
    matrixTD <- head(matrixTD, -N)
  matrixTD[matrixTD>=1] <- 1
  tam <- matrixTD %*% t(matrixTD)
  return (tam)
})
```

A `memoise` segítségével érhetjük el, hogy változatlan bemenő értékekre ne fusson le újból a függvény, hanem csak keresse elő az adott bemenethez tartozó, előzetesen lementett visszatérési értéket.

A függvény három paraméterrel rendelkezik:

- `adta`: Az adathalmazt adja meg, melyből az asszociációs mátrixot szeretnénk létrehozni.
- `corlimit`: Az asszociációs hálózat korrelációs határát adja meg.
- `maxword`: Az asszociációs hálózatban szereplő szavak maximális számát adja meg.

Először lecsökkentjük a későbbi ábrázolásban szereplő szavakat a beállított korrelációs határnak megfelelően. Kivesszük az adathalmaz minden olyan szavát, mely ez alatt az érték alatt van.

Egy mátrixot készítünk belőle, aminek levágjuk az utolsó N sorát a maximális szó számnak megfelelően.

Végül a mátrixot megszorozzuk a saját transzponáltjával, hogy kapjunk egy olyan diagonális mátrixot, melyben a főátló elemi 1-ek.

4.4. A szófelhő elkészítése

```
myWordCloud <- function(myFrame){
  myPlot <- ggplot(myFrame,aes(myFrame[,3],myFrame[,4],
                 label = rownames(myFrame))) +
                 geom_point(color= "#00FF00", alpha=0.45, size=9)

  myPlot <- myPlot + geom_text(aes(size=freq)) +
            scale_size(range=c(2.5,20))

  myPlot <- myPlot + theme(legend.position="none",
                           panel.grid.major=element_blank(),
                           panel.grid.minor=element_blank(),
                           panel.border=element_blank(),
                           panel.background=element_blank(),
                           axis.title.x=element_blank(),
                           axis.text.x=element_blank(),
                           axis.ticks.x=element_blank(),
                           axis.title.y=element_blank(),
                           axis.text.y=element_blank(),
                           axis.ticks.y=element_blank())
  myPlot
}
```

A függvény paramétere:

- `myFrame`: Megadja, hogy melyik Data Frame-ből szeretnénk elkészíteni a kirajzolandó szófelhőt.

Beállítjuk, hogy a Data Frame harmadik és negyedik oszlopa szerint szeretnénk ábrázolni az értékeket. (A harmadik oszlopot X koordináta értékkel, a negyedik oszlopot pedig Y koordináta értékkel töltöttük fel korábban.) Az ábrázolt pontokhoz tartozó feliratok/címkék a táblázat nulladik oszlopában találhatóak. Végül beállítjuk a pontok színét, áttetszőségét, és területét.

Beállítjuk továbbá a pontokhoz tartozó címkék méretét, hogy az a kifejezések előfordulásával legyen arányban, ami egy 2.5-20 skálán helyezkedik el.

Az utolsó lépésben pedig kivesszük az ábrából a fő- és segédvonalakat, a keretet, a hátteret, az X és Y tengelyeket, az X és Y tengelyek feliratait, valamint a tengelyek beosztásait.

4.5. Beolvasott fájl kiterjesztésének meghatározása

```
substrRight <- function(x, n){
  substr(x, nchar(x)-n+1, nchar(x))
}
```

A függvény két paraméterrel rendelkezik:

- x: A string, amely utolsó N darab karakterét szeretnénk meghatározni.
- n: Megadja, az utolsó karakterek számát

A `substr` függvény segítségével, az adott stringből kiszedhető egy string, melynek megadjuk a kezdő és vég pozícióját az eredeti stringben. Az `nchar` függvény az adott string hosszával tér vissza.

Például:

```
>x <- "teststring"
>substr(x, 2, 4)
[1] "est"
>
>substr(x, 2, nchar(x)
[1] "eststring"
>
>substr(x, nchar(x)-2, nchar(x))
[1] "ing"
```

4.6. Interakciós részek

Ide tartozik az egérkattintás koordinátáinak vizsgálata, valamint a koordinátához tartozó szó és az előfordulási szám keresése.

4.6.1. 1.5 sugarú környezet

Ahhoz, hogy a szófelhő megfelelő szavait vissza tudjuk kapni, szükség van az ábrázolt pontok egy bizonyos környezetének az elfogadásához is. Így például nem szükséges pontosan a P(5;5) pontra kattintani. Elégséges a P pont 1.5 sugarú környezetét eltalálni.

```
coordrange <- function(A,B,C,D){
  if(A+1.5 > B && A-1.5 < B && C+1.5 > D && C-1.5 < D) return (T)
  else return (F)
}
```

A függvény négy paraméterrel rendelkezik:

- A: Az egérkattintáshoz tartozó X koordináta.
- B: Az X koordináta, amely szavát szeretnénk visszakapni.
- C: Az egérkattintáshoz tartozó Y koordináta.
- D: Az Y koordináta, amely szavát szeretnénk visszakapni.

Ha az egérkattintáshoz tartozó koordináta benne van egy szóhoz tartozó koodrdináta 1.5 sugarú környezetében, akkor a függvény visszatérési értéke: TRUE

Ellenkező esetben: FALSE

4.6.2. A koordinátához tartozó szó

```
word_str <- function(e,myFrame){
  valid = F
  if(is.null(e)) return("NULL\n")
  if(!is.null(myFrame)){
    for (i in 1:nrow(myFrame)){
      if(coordrange(round(e$x,1),myFrame[i,3],
                    round(e$y,1),myFrame[i,4])){
        valid = T
        rowNumb = i
      }
    }
  }
  else return ("NULL\n")
```

A függvény két paraméterrel rendelkezik:

- e: Az egérkattintáshoz tartozó koordináták.
- myFrame: A táblázat, melyben keressük a koordinátákhoz tartozó szavakat.

Létrehozunk egy valid nevű változót. Az értéke akkor változik, ha az egérkattintáshoz tartozó koordinátáknak találtunk megfelelő koordinátákat a táblázatban.

Ha létezik a táblázat, amiben keresni szeretnénk, akkor elindulunk az első sorából. Minden egyes sornál megnézzük, hogy a kattintással szerzett koordinátapár benne van-e a táblázatban szereplő koordinátapárok valamelyikének az 1.5 sugarú környezetében. Ha találtunk ilyet, akkor megjegyezzük az adott sor számát és átállítjuk a `valid` segédváltozót `TRUE`-ra.

```
    if(valid == T){
      return (myFrame[rowNumb,1],
    }
    else(return("NULL\n"))
  }
```

Ha a `valid` segédváltozó értéke `TRUE`, akkor a megjegyzett sorból kimásoljuk a koordináta szavát.

A függvény visszatérési értéke `NULL`, ha:

- Az egérrel nem kattintottunk sehova.
- A táblázat nem létezik, amiben keresni szeretnénk.
- A táblázatban nem találtunk a feltételeknek megfelelő koordinátákat.

4.6.3. Adott szóhoz tartozó előfordulási szám

```
freq_str <- function(word,myFrame){
  valid = F
  if(is.null(word)) return("NULL")
  if(!is.null(myFrame)){
    for (i in 1:nrow(myFrame)){
      if(word == myFrame[i,1]){
        valid = T
        rowNumb = i
      }
    }
```

A függvény két paraméterrel rendelkezik:

- `word`: A táblázatból keresendő szó.
- `myFrame`: A táblázat, melyben keressük a szóhoz tartozó előfordulási számot.

Létrehozunk egy `valid` nevű változót. Az értéke akkor változik, ha a táblázatban megtaláltuk a kereset szót.

Ha létezik a táblázat, amiben keresni szeretnénk, akkor elindulunk az első sorából. Minden egyes sornál megnézzük, hogy a keresett szó megegyezik-e, az aktuális sorban lévő szóval. Ha találtunk ilyet, akkor megjegyezzük az adott sor számát és átállítjuk a `valid` segédváltozót `TRUE`-ra.

```
      }
    }
    else return ("NULL")
    if(valid == T){
      paste0(myFrame[rowNumb,2])
      return (myFrame[rowNumb,2])
    }
    else(return("NULL"))
  }
```

Ha a `valid` segédváltozó értéke `TRUE`, akkor a megjegyzett sorból kimásoljuk a szóhoz tartozó előfordulási számot.

A függvény visszatérési értéke `NULL`, ha:

- Nincs keresendő szó.
- A táblázat nem létezik, amiben keresni szeretnénk.
- A táblázatban nem találtunk a keresett szót.

5. server.R

Két fő részre bontható:

- NCBI adatok feldolgozása és ábrázolása.
- Kiválasztott fájl feldolgozása és ábrázolása.

Mindkét rész folyamata azonos:

1. Először a korpuszműveleteket hajtjuk végre, rendezett Data Framet készítünk.
2. Létrehozzuk, majd ábrázoljuk a szófelhőt.
3. A szófelhő interakciós funkciói (nagyítás, kattintási információ).
4. Létrehozzuk, majd ábrázoljuk az asszociációs hálózatot.

5.1. NCBI korpuszműveletek

A korpuszműveletek közé tartozik a szöveg tisztítása, a kifejezés-dokumentum mátrix készítése és a Data Frame készítése, amiből a szövegfelhő lesz ábrázolható.

5.1.1. Nagyításhoz használt segédváltozók

```
ranges <- reactiveValues(x = NULL, y = NULL)

ranges5 <- reactiveValues(x = NULL, y = NULL)
```

Nagyításhoz használt segédváltozók. A `reactiveValues` függvény visszatér a benne eltárolt értékekkel. Ezeket a koordinátapárokat adjuk át később a szófelhő ábrázolásánál. Az értékeit a szófelhőn lévő, egérrel való kijelölési területből kapja. Alapértelmezett esetben az X és Y pontok értéke NULL.

5.1.2. Rendezett kifejezés-dokumentum mátrix az NCBI adatbázisból

```
terms <- reactive({
  input$update

  isolate({
    withProgress({
      setProgress(message = "Szófelhő előállítása:")
      res <- EUtilsSummary(input$text,db=input$selection,
                           retmax = input$num,
                           mindate=input$date)
      summary(res)
      fetch <- EUtilsGet(res)
      PMID(fetch)
      adat <- AbstractText(fetch)
      adat <- myTermDocumentMatrix(adat)
      getTermMatrix(adat)
    })
  })
})
```

A „terms” nevű reaktív. Akkor hajtódik végre, ha az update gombra kattintunk. Ezután megjelenik a „Szófelhő előállítása:” üzenet.

Az `EUtilsSummary` paranccsal összefoglaló információkat készítünk a lekérdezéshez. Tartalmazza a keresendő kifejezést, az NCBI-ról kiválasztott forrást, a maximálisan jegyzendő szavak számát és, hogy mi legyen a minimum dátuma a vizsgálandó dokumentumoknak.

Az `EUtilsGet` segítségével letöltjük a lekérdezés eredményét. Kiszedjük a Medline objektumból a PMID[10]-t. Ennek segítségével érhetjük el, mely dokumentumokat szeretnénk feldolgozni. Kiszedjük belőlük az absztrakt szövegeket, ami megadja a szövegkorpuszt.

Az eredményből egy kifejezés-dokumentum mátrixot készítünk, végül rendezett mátrixszá alakítjuk.

5.1.3. Data Frame az NCBI adatbázisból

```
DataFrameFromNCBI <- reactive({
  myTDM <- terms()
  myFrame <- getMyDataFrame(myTDM,input$max,input$freq)
  return (myFrame)
})
```

Data Frame készítése az NCBI-ból kinyert rendezett mátrixból.

5.2. NCBI szófelhő

```
output$plot <- renderPlot({
  myFrame <- DataFrameFromMEDHUB()
  if(is.null(terms())) return (NULL)
  myWordCloud(myFrame) + coord_cartesian(xlim = ranges$x,
                                         ylim = ranges$y)
})
```

Az NCBI-ból kinyert Data Frame-et kirajzoljuk a `myWordCloud` segítségével, majd beállítjuk a megjelenítendő területet X és Y határai segítségével.

Ha nem készült el a kifejezés-dokumentum mátrix, akkor a visszatérési érték NULL, így az ábrázolási terület üres marad.

[10] PMID – A PubMed saját kézirat/dokumentum azonosítója.

5.3. NCBI interakciók

Az interakciók közé tartozik a szófelhő nagyítása, valamint az egérkattintáshoz tartozó kifejezés kinyerése.

5.3.1. Az NCBI szófelhő nagyítása

```
observeEvent(input$plot_dblclick, {
  brush <- input$plot_brush
  if (!is.null(brush)) {
    ranges$x <- c(brush$xmin, brush$xmax)
    ranges$y <- c(brush$ymin, brush$ymax)

  } else {
    ranges$x <- NULL
    ranges$y <- NULL
  }
})
```

Az `observeEvent` segítségével beállítjuk, hogy dupla kattintás esetén vizsgáljuk meg, van-e kijelölt terület az NBCI adatbázisból ábrázolt szófelhőn. Ha van, akkor a kijelölt terület határkoordinátáit mentsük el a nagyításhoz használt változókba. Ha nincs kijelölt terület, akkor a segédváltozók értékét állítsuk NULL-ra.

5.3.2. Az NCBI szófelhőhöz tartozó kattintási információk átadása

```
output$info <- renderText({
  myFrame <- DataFrameFromNCBI()
  selectedWord <- word_str(input$plot_click,myFrame)

  paste0(
    "word: ", selectedWord,
    "\nfreq: ", freq_str(selectedWord,myFrame)
  )
})
```

Segítségével átadjuk az `info` kimenetre az NCBI szófelhőhöz tartozó kattintási információkat. Megadjuk, hogy melyik Data Frame-ben szeretnénk keresni, majd kimásoljuk a megtalált cella tartalmát, és a hozzá tartozó előfordulási számot.

5.4. NCBI asszociációs hálózat

Az asszociációs hálózathoz szükség van egy szomszédsági mátrixra. Ennek előállítása után kezdhetjük a kirajzolás beállításait.

5.4.1. Asszociációs hálózat az NCBI adatbázisból

```
boole <- reactive({
  input$booleButton
  isolate({
    withProgress({
      setProgress(message = "Network structure")
      res <- EUtilsSummary(input$text,db=input$selection,
                           retmax = input$num,
                           mindate=input$date)
      summary(res)
      fetch <- EUtilsGet(res)
      PMID(fetch)
      adat <-AbstractText(fetch)
      adat <- myTermDocumentMatrix(adat)
      getNetwork(adat,input$corlim,input$max)
    })
  })
})
```

A „bool” nevű reaktív akkor hajtódik végre, ha az „Network” gombra kattintunk. Ezután megjelenik a „Network structure ” üzenet.

Az `EUtilsSummary` paranccsal összefoglaló információkat készítünk a lekérdezéshez. Tartalmazza a keresendő kifejezést, az NCBI-ból kiválasztott forrást, a maximálisan megjelenítendő találatok számát és, hogy mi legyen a keresés idő intervalluma.

Az `EUtilsGet` segítségével letöltjük a teljes lekérdezés eredményét. Kiszedjük a Medline objektumból a PMID-t. Ennek segítségével érhetjük el, mely dokumentumokat szeretnénk feldolgozni. Kiszedjük belőlük az absztrakt szövegeket, ami megadja a szövegkorpuszt.

Az eredményből egy kifejezés-dokumentum mátrixot készítünk, végül szomszédsági mátrixszá alakítjuk.

5.4.2. Asszociációs hálózat átadása az NCBI-ból

```
output$plot3 <- renderSvgPanZoom({
  ccc <- boole()
  matrix.g <- graph.adjacency(ccc, weighted = T,
                              mode = "undirected")
  matrix.g <- simplify(matrix.g)
```

A `plot3` nevű kimenethez hozzárendeljük az asszociációs hálózatot.

Miután a `boole` segítségével megkaptuk a kívánt asszociációs hálózatot, egy szomszédsági gráfot készítünk belőle úgy, hogy az összeköttetések súlyozva legyenek (ha `weighted = F`; akkor csak egy számérték jelenik meg a két kifejezés között, így viszont az összeköttetés vastagsága változik). Az összeköttetések módját irányítatlannak választjuk, majd a `simplify` segítségével eltávolítjuk belőle a hurkokat és a gráf párhuzamos éleit.

```
  V(matrix.g)$label <- V(matrix.g)$name
  V(matrix.g)$degree <- degree(matrix.g)
  V(matrix.g)$label.cex <- 2.2 *
                           V(matrix.g)$degree/
                           max(V(matrix.g)$degree)+.2
  V(matrix.g)$label.color <- rgb(0.6,0,0)
  V(matrix.g)$frame.color <- NA
```

A gráf csomópontjainak címkéit átírjuk a csomópontok nevére, valamint a csomópontok fokát beállítjuk a mátrixból kinyert fokokra. A következő lépésben a csomópontok címkéit a csomópontok fokával arányos betűnagyságura, a szöveg színét pedig 0.6/0/0-ára (sötétpiros) állítjuk, valamint kiszedjük a csomópontok keretét.

```
  egam <- (log(E(matrix.g)$weight) +
           .4)/max(log(E(matrix.g)$weight) + .4)
  E(matrix.g)$color <-rgb(.5, .5, 0, egam)
  E(matrix.g)$width <- egam
```

A gráf éleinek vastagságát beállítjuk a súlyozásukkal arányosan. Színüknek pedig a 0.5/0.5/0-át választjuk (zöldes barna).

```
    layout1 <- layout.fruchterman.reingold(matrix.g)
    svgPanZoom(
      svglite:::inlineSVG(
      plotgraph <- plot(matrix.g, layout = layout1)
      )
    )
  })
```

A csomópontok és az összekötő élek beállítása utána, a `layout.fruchterman.reingold` függvény segítségével, a kétoszlopos mátrix koordinátáit egy külön változóba másoljuk, hogy a `plot` ki tudja rajzolni, majd átültetjük svg formátumba a `plot`-tal ábrázolt képet. Így az nagymértékben, torzulásmentesen nagyítható lesz

5.5. Fájl kiválasztása, beolvasása

```
  myChoosedFile <- reactive({
    # Change when the "update" button is pressed...
    input$fileupdate
    # ...but not for anything else
    isolate({
      withProgress({
        setProgress(message = "Processing corpus...")
        frompdf = 0
        TEXTFILE <- input$infile
        if(is.null(TEXTFILE)) return (NULL)
```

A „fileupdate” névű gombra kattintva aktiválódik a reaktív. Amíg tart a folyamat, megjelenítünk egy „Processing corpus...” feliratot.

A fileInput segítségével kiválasztható, hogy milyen fájlt szeretnénk beolvasni.

Ha még nem választottunk semmit, akkor a visszatérési érték NULL

A fileInput egy ideiglenes mappába másolja az általunk kiválasztott fájlt. Majd a hozzá tartozó információkat a következőképpen tárolja:

	name	**size**	**type**	**datapath**
1	A kiválasztott fájl	A fájl mérete	A fájl típusa	A fájl elérési útvonala

3. táblázat: *fileInput adatok*

```
          attr = substrRight(TEXTFILE[1,1], 3)
          if(attr == "pdf"){
            frompdf = 1
            uri = TEXTFILE[1,4]
            myFile <- readPDF(engine = "xpdf",
                              control=list(text = "-layout"))
                             (elem = list(uri = uri),
                              language = "en",
                              id = "id1")
          }
```

A `substrRight` függvénnyel megvizsgáljuk a fájl kiterjesztését.

Ha pdf, akkor a `frompdf`-et átállítjuk 1-re, kiszedjük a fájl elérési útvonalát, majd a `readPDF` függvényt használva, a külső pdf olvasóval nyitjuk meg és olvassuk be.

```
          if(attr == "txt" || attr == "csv"){
            frompdf = 0
            con <- file(TEXTFILE[1,4])
            myFile <- readLines(con,
                                encoding="UTF-8")
            close(con)

          }
          myFile <- myTermDocumentMatrix(myFile,frompdf,uri)
          return (myFile)
        })
      })
    })
```

Ha txt vagy csv, akkor a `frompdf` marad 0, kiszedjük a fájl elérési útvonalát, majd soronként beolvassuk.

Végül kifejezés-dokumentum mátrixot készítünk a beolvasott szövegből, ami a reaktív visszatérési értéke is lesz.

5.6. Kiválasztott fájlhoz tartozó korpuszműveletek

A korpuszműveletek közé tartozik a szöveg tisztítása, a kifejezés-dokumentum mátrix készítése és a Data Frame készítése, amiből a szövegfelhő lesz ábrázolható.

5.6.1. Data Frame készítése a kiválasztott fájlból

```
DataFrameFromFile <- reactive({
  myTDM <- myChoosedFile()
  if(is.null(myTDM)) return (NULL)
  myTDM <- getTermMatrix(myTDM)
  myFrame <- getMyDataFrame(myTDM,input$max,input$freq)
  return (myFrame)
})
```

A „DataFrameFormFile” nevű reaktív. Először a kiválasztott fájlból egy kifejezés-dokumentum mátrix készül. Ha ezt nem sikerül létrehozni (még nem volt fájl kiválasztva), akkor a visszatérési értékünk NULL. Ellenkező esetben rendezzük a kifejezés-dokumentum mátrixot és elkészítjük belőle a Data Frame-et. Végül visszatérünk vele.

5.7. Szófelhő a kiválasztott fájlból

Ahhoz, hogy a kiválasztott fájlból és az NCBI adatbázisból készült szófelhőt párhuzamosan tudjuk kezelni/megjeleníteni, szükségünk van egy új ’plot’ kimenetre.

5.7.1. Szófelhő készítése a kiválasztott fájlból

```
wordc <- reactive({

  myFrame <- DataFrameFromFile()

  if(!is.null(myFrame)){
    myWordCloud(myFrame)

  }
  else {
    return (NULL)
  }
})
```

Felhasználjuk a Data Frame készítő függvényt, majd a visszatérési értékét, ha van, átadjuk a szófelhőt készítő függvénynek. Ha a `DataFrameFromFile` függvényünk visszatérési értéke NULL volt, akkor itt is NULL-al térünk vissza.

5.7.2. Szófelhő átadása a kiválasztott fájlból

```
output$plot5 <- renderPlot({
  if(!is.null(wordc())){

    temp <- wordc()
    temp <- temp + coord_cartesian(xlim = ranges5$x,
                                   ylim = ranges5$y)
    temp
  }
  else{
    return (NULL)
    }
})
```

A kiválasztott fájlból kinyert Data Frame-et kirajzoljuk a `wordc` segítségével, majd beállítjuk a megjelenítendő területet X és Y határai segítségével.

Ha a `wordc` visszatérési értéke NULL, akkor itt is NULL értékkel térünk vissza, így az ábrázolási terület üres marad.

5.8. Interakciók a szófelhőhöz

Mivel a két szófelhő ábrázolása párhuzamosan történik, ezért meg kell különböztetni, hogy melyik szófelhőt szeretnénk nagyítani, melyikből szeretnénk kinyerni a kattintás információit.

5.8.1. A kiválasztott fájlból készült szófelhő nagyítása

```
observeEvent(input$plot5_dblclick, {
  brush <- input$plot5_brush
  if (!is.null(brush)) {
    ranges5$x <- c(brush$xmin, brush$xmax)
    ranges5$y <- c(brush$ymin, brush$ymax)

  } else {
    ranges5$x <- NULL
    ranges5$y <- NULL
  }
})
```

Az `observeEvent` segítségével beállítjuk, hogy dupla kattintás esetén vizsgáljuk meg, van-e kijelölt terület a kiválasztott fájlból készült szófelhőn. Ha van, akkor a kijelölt terület határkoordinátáit mentsük el a nagyításhoz használt változókba. Ha nincs kijelölt terület, akkor a segédváltozók értékét állítsuk NULL-ra.

5.8.2. A kiválasztott fájlhoz tartozó kattintási információk átadása

```
output$info5 <- renderText({
  myFrame <- DataFrameFromFile()
  selectedWord <- word_str(input$plot5_click,myFrame)
  paste0(
    "word: ", word_str(input$plot5_click,myFrame),
    "\nfreq: ", freq_str(selectedWord,myFrame),
  )
})
```

Segítségével átadjuk az `info5` kimenetre a fájlból kiválasztott szófelhőhöz tartozó kattintási információkat. Megadjuk, hogy melyik Data Frame-ben szeretnénk keresni. Majd kimásoljuk a megtalált cella tartalmát, és a hozzá tartozó előfordulási számot.

5.9. A kiválasztott fájl asszociációs hálózata

Mivel a két szófelhő ábrázolása párhuzamosan történik, ezért meg kell különböztetni, hogy melyik szófelhőből szeretnénk az asszociációs hálózatot létrehozni.

5.9.1. Asszociációs hálózat készítése a kiválasztott fájlból

```
NetworkGraph <- reactive({

  myTD <- myChoosedFile()
  myFrame <- DataFrameFromFile()
  if(is.null(myChoosedFile())) return (NULL)
  myTD1 = removeSparseTerms(myTD, 0.993)
  matrixTD <- as.matrix(myTD1)
```

A „NetworkGraph” nevű reaktív. A kiválasztott fájlból készült asszociációs hálózat készítését és ábrázolását végzi el. A kiválasztott fájlból elkészíti a kifejezés-dokumentum mátrixot és a Data Frame-et.

Ha nem választottunk még ki semmilyen fájlt, a visszatérési értéke NULL, hogy az ábrázolási terület üres maradjon.

Eltávolítjuk a korpuszból a 0.993 korrelációs határ alatti kifejezéskét, majd az így redukált eredményt mátrixszá alakítjuk.

```
N <- nrow(matrixTD)-input$max
if(nrow(matrixTD)>input$max)
  matrixTD <- head(matrixTD, -N)
matrixTD[matrixTD>=1] <- 1
```

Ahogy a Data Frame-nél és a másik asszociációs hálózatnál is tettük, levágjuk az adathalmazunk utolsó N darab sorát, a maximum számot beállító csúszkának megfelelően.

```
startWord <- word_str(input$plot5_click,myFrame)
if (is.null(input$plot5_click)){
  return (NULL)
}
myAssoc <- findAssocs(myTD1,c(startWord), c(0.1))
dtassoc <-as.data.frame(myAssoc)
dtassoc <-as.matrix(dtassoc)
dtassoc <-dtassoc %*% t(dtassoc)
```

Kattintással kinyerjük az asszociációs hálózat kívánt kiindulási kifejezését. Ha nem kattintottunk rá egyik szóra sem, akkor NULL értékkel térünk vissza, hogy a az ábrázolási terület üres maradjon.

Ha kiválasztottuk az egyik kifejezést, akkor `findAssocs` segítségével felállítjuk a kifejezéshez tartozó asszociációs listát. Ahhoz, hogy ezt a listát ábrázolni tudjuk, egy diagonális mátrixszá kell alakítanunk. Ezt úgy tehetjük meg, hogy a listából készítünk egy Data Frame-et, abból pedig egy mátrixot. Ha ezzel megvagyunk, már csak meg kell szorozni a kapott mátrixot a saját transzponáltjával.

```
matrix.g <- graph.adjacency(dtassoc, weighted = T,
                            mode = "undirected")
matrix.g <- simplify(matrix.g)
```

Miután megkaptuk a kívánt mátrixot, egy szomszédsági gráfot készítünk belőle úgy, hogy az összeköttetések súlyozva legyen (ha `weighted = F`; akkor csak egy számérték jelenik meg a két kifejezés között, így viszont az összeköttetés vastagsága változik). Az összeköttetések módját irányítatlannak választjuk, majd a `simplify` segítségével eltávolítjuk belőle a hurkokat és a gráf párhuzamos éleit.

```
      V(matrix.g)$label <- V(matrix.g)$name
      V(matrix.g)$degree <- degree(matrix.g)
```

A gráf csomópontjainak címkéit átírjuk a csomópontok nevére, valamint a csomópontok fokát beállítjuk a mátrixból kinyert fokokra.

```
      TMP <-2.2 * V(matrix.g)$degree/max(V(matrix.g)$degree) + .2
      V(matrix.g)$label.cex <- A
      V(matrix.g)$label.color <- rgb(0.6,0,0)
      V(matrix.g)$frame.color <- NA
```

A gráf csomópontjainak címkéit a csomópontok fokával arányos betűnagyságura, a szöveg színét pedig 0.6/0/0-ára (sötétpiros) állítjuk, valamint kiszedjük a csomópontok keretét.

```
      TMP <- log(E(matrix.g)$weight)
      egam <- (TMP) + .4)/max(TMP) + .4)
      E(matrix.g)$width <- egam
      E(matrix.g)$color <-rgb(.5, .5, 0, egam)
```

A gráf éleinek vastagságát beállítjuk a súlyozásukkal arányosan. Színüknek pedig a 0.5/0.5/0-át választjuk (zöldes barna).

```
      layout1 <- layout.fruchterman.reingold(matrix.g)
      plotgraph <- plot(matrix.g, layout = layout1)
      return (plotgraph)
    })
```

A csomópontok és az összekötő élek beállítása utána, a `layout.fruchterman.reingold` függvény segítségével, a kétoszlopos mátrix koordinátáit egy külön változóba másoljuk, hogy a `plot` ki tudja rajzolni.

5.9.2. Asszociációs hálózat átadása kiválasztott fájlból

```
  output$graph <- renderSvgPanZoom({
   if(is.null(wordc())) return (NULL)
   svgPanZoom(
     svglite:::inlineSVG(NetworkGraph())
   )
 })
```

A `graph` nevű kimenethez hozzárendeljük az asszociációs hálózatot. Ez az `svgPanZoom` segítségével történik. Ez egyszerűen csak svg formátumba ülteti át a `plot`-tal ábrázolt képet. Így az nagymértékben, torzulásmentesen nagyítható lesz.

5.10. Mentés pdf vagy png fájlként

```
    output$down <- downloadHandler(
      filename =  function(){
        paste("test", input$radio, sep="")
      },
      content = function(vmi) {

        save <- wordc()

        if(input$radio == ".png"){
          png(vmi)
          print(save, xlim=c(0,100), ylim=c(0,100))
        }
        if(input$radio == ".pdf")
          pdf(vmi)


        dev.off()
      }
```

A `downloadHandler` segítségével kezeljük a mentést. Alapértelmezetten a menteni kívánt fájl neve „test” lesz, a kiterjesztése pedig a `radioButton` segítségével kiválasztva vagy .png, vagy .pdf. Sajnálatos módon `downloadHandler` nem tudja megfelelően kezelni a reaktívokat.

Pdf esetén megnyitja a lementeni kívánt fájlt, majd a pdf olvasó segítségével fejezhető be a mentés.

Png esetén, egy üres kép mentődik le. Ezt kiküszöbölve a `print` paranccsal lett lementve a png. Viszont itt nincs lehetőség útvonalválasztásra, automatikusan oda mentődik a kép, ahonnan a program fut.

6. Összefoglalás

A szakdolgozatom végén, szeretnék kitérni a kitűzött feladatok megvalósításának sikerességére. Bemutatnám az alkalmazás használatának folyamatát, valamint szeretnék kitérni a programmal kapcsolatos folytatási lehetőségekre.

6.1. Szövegbányász adatstruktúrák

Feladatom elvégzése során megismerkedtem a szövegbányászati folyamattal. Megtanultam, hogy a korpusz a feldolgozandó dokumentumok összessége, hoztam létre dokumentum-kifejezés mátrixokat és transzponáltjaikat (kifejezés-dokumentum mátrix), valamint az elkészített mátrixokból felépítettem a saját Data Frame-em, aminek segítségével szófelhőket ábrázoltam.

6.2. Adattisztítás

A szövegbányászati feladatok során egy négylépéses tisztítási folyamatot használtam, mely lépései:

- A korpuszban található nagybetűk kicserélése kisbetűkre
- Írásjelek eltávolítása
- Számok eltávolítása
- Az úgynevezett stopszavak eltávolítása (névelők, kötő szavak, névmások, stb.)

6.3. Interaktív szófelhő és szó korreláció

A tm, ggplot2 és ggrepel csomagok segítségével elvégeztem egy interaktív szófelhő megtervezését, amely reagál az egér akciókra. Így kattintással képes visszaadni az egér alatt lévő kifejezést, valamint lehetséges benne a nagyítás. A címkék egymás fedésének elkerülése még nem működik rendesen. Azt sikerült megoldani, hogy ne legyen két azonos koordinátájú címke, viszont magasságuk és szélességük egyaránt függ az előfordulások számától és a kifejezés hosszúságától.

A szókorreláció vizualizációja idő hiányában nem lett befejezve. Az általunk kiválasztott fájlra működik. Itt kiválasztható, hogy melyik kifejezésből induljon az asszociációs hálózat, valamint a vizualizáció svg alapú, így nagymértékben, torzulás nélkül nagyítható. Viszont a folyamatot nem volt időm átemelni az NBCI adatbázis korpuszára. Onnan csak a korpusz legyakrabban előforduló szavára készül el az asszociációs hálózat, kezdőszó még nem választható interaktív módon.

6.4. Medline adatbázis

A Medline adatbázisból képes a program szófelhőt előállítani, és a korpusz leggyakoribb szavából egy asszociációs hálózatot kirajzolni. Ennek elvégzése a RISmed csomag segítségével történik. A csomag képes a PubMed keresőnek átadni a keresési paramétereket, majd kinyerni a keresés eredményeit, illetve letölteni és feldolgozni az absztraktokat.

6.5. GUI: Grafikus felhasználói felület

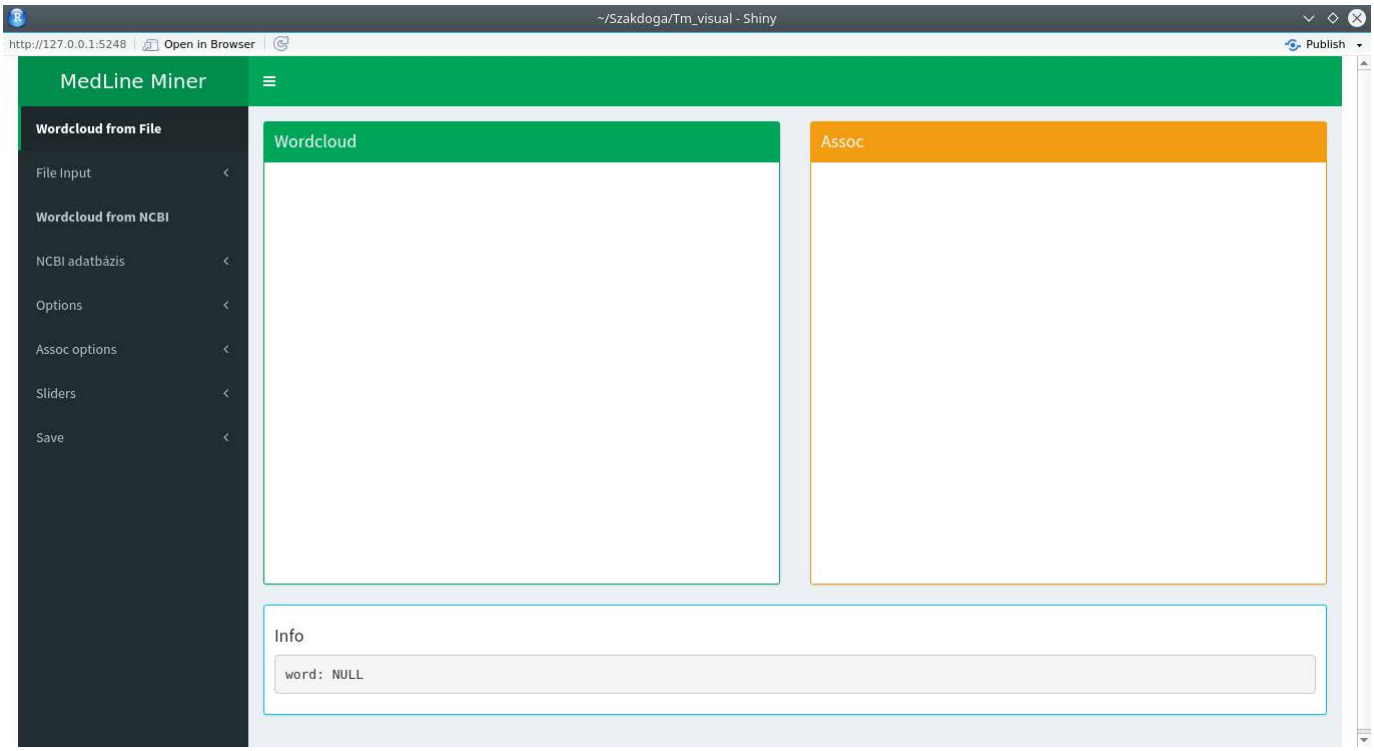
A grafikus felhasználói felületen helyeztem el csúszkákat, gombokat és bemeneti mezőket. A csúszkákkal a maximálisan megjelenítendő kifejezések száma, a minimális elfordulási mérték, és a korrelációs határok állíthatók. A gombok a beállított értékek megerősítésére, valamint a szófelhő és az asszociációs hálózat újbóli kirajzolására szolgálnak. A bemeneti mezők pedig a kiválasztandó fájl kereséséhez, az NCBI adatbázisban való kereséshez, és az asszociációs hálózat kiindulási szavának beállításához használhatóak.

6.6. A program használata

A következőben szeretném bemutatni a program használatát. Mind a fájl kiválasztását és feldolgozását, mind az NCBI adatbázisól való feldolgozást.

Az általam kiválasztott fájllal szeretném kezdeni, ami: A *The Hitch Hiker's Guide to the Galaxy* txt változata.

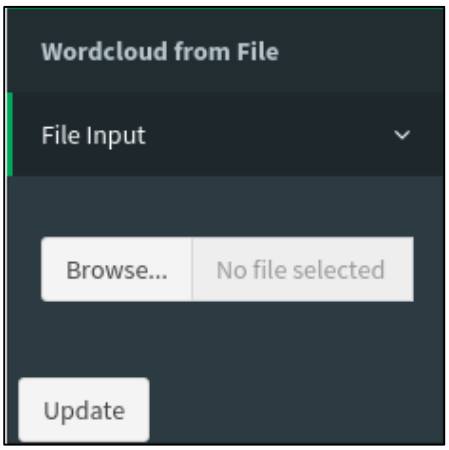
A program megnyitásakor a következő kép fogad bennünket:



4. ábra: *A program kezdő képe*

A bal felső sarokban látható a cím. Alatta az interaktív menü lista. A vastagon szedett menük kivételével mindegyik legördülő menü. A két vastagon szedett pedig arra alkalmas, hogy fájlból vagy az NCBI adatbázisból kirajzolt szófelhőt és asszociációs hálózatot jelenítse meg.

Mivel még nem választottunk ki semmi fájlt, ezért a két box tartalma üres, illetve a kiválasztott szó értéke NULL.



5. ábra: *File Input menü*

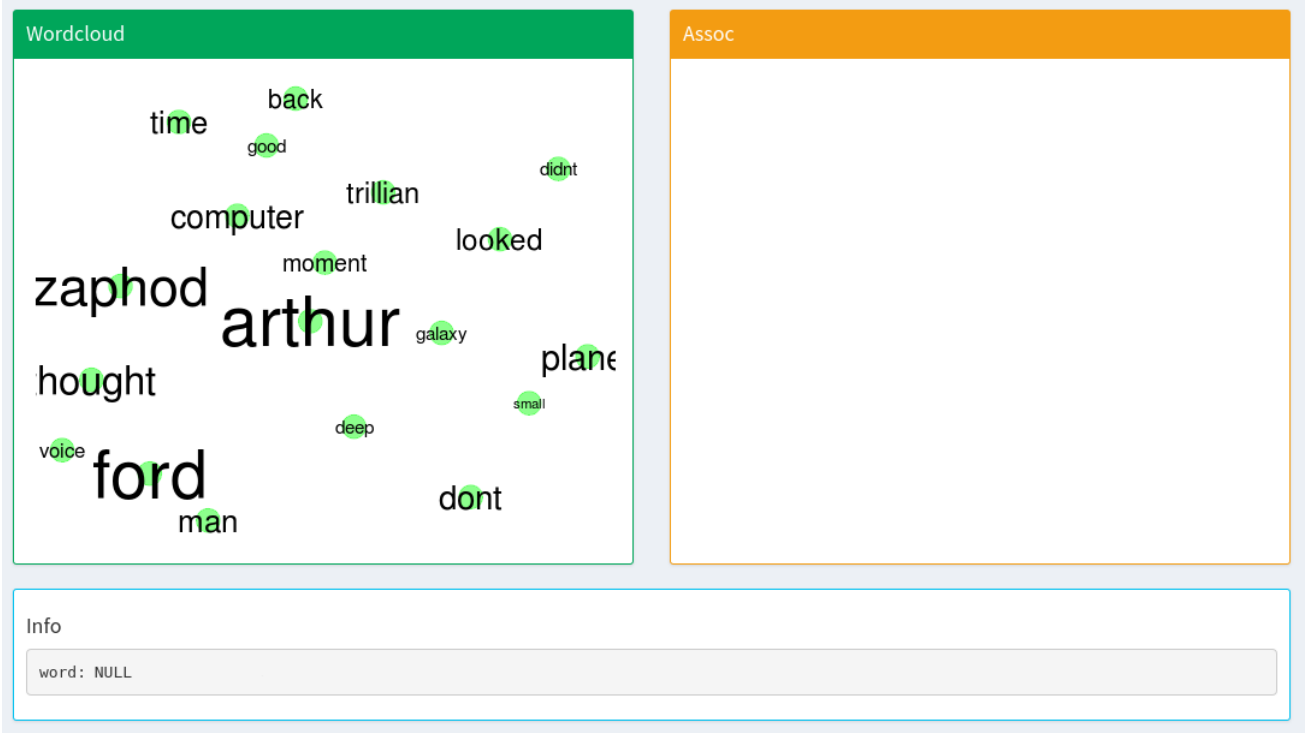
Fájl kiválasztásához, a „File Input” legördülő menüre kell kattintanunk.

Ahogy a fül lenyílik, látható lesz egy „Browse…” feliratú gomb, mellette a kiválasztott fájl neve, alattuk pedig az „Update” gomb.

Kattintsunk a „Browse…” gombra. Ennek hatására előugrik egy ablak, melyben kikereshetjük a kívánt dokumentumot.

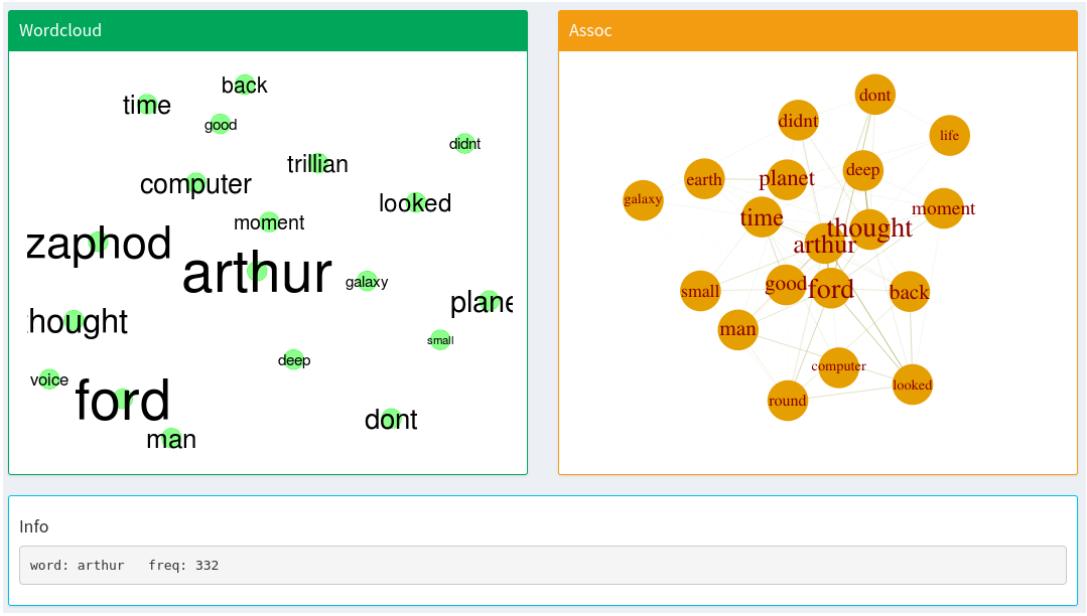
Végül kattintsunk az „Update” gombra. Így a program elkezdi a kiválasztott fájl feldolgozását.

Egy kis idő múlva meg is jelenik a létrehozott szófelhő. Mikor innen nincs szó kiválasztva, nincs miből készíteni az asszociációs hálózatot, ezért itt még nem jelenik meg semmi.



6. ábra: *A szófelhő*

A program mostani állapotában a kirajzolt szavak közepénél lévő zöld körök jelölik a kifejezések 1.5 sugarú köryezetét. Ezekre a zöld körökre kattintva tudjuk a megfelelő kifejezést kinyerni. Vigyük az egeret a szófelhő fölé, majd kattintsunk az egyik ilyen zöld körre.



7. ábra: *Szófelhő és asszociációs hálózat*

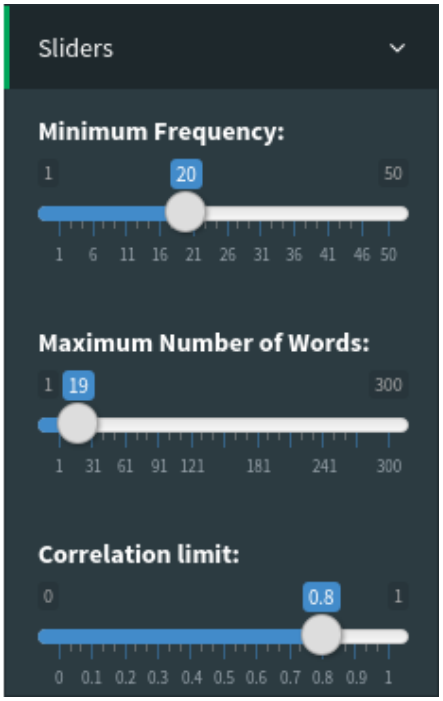
Az általam kiválasztott szó, az „arthur” volt. Ennek hatására kirajzolódik az asszociációs hálózat. Az „Info” box-ban pedig láthatóvá válik, melyik szó lett kiválasztva és megjelenik vele együtt az is, hányszor fordul elő a kifejezés a dokumentumban.

A csúszkákkal való beállításokhoz kattintsunk a „Sliders” legördülő menüre.

Itt beállíthatók:

- a szófelhőbe kerülő kifejezések minimális előfordulási száma,
- a szófelhőben és az asszociációs hálózatban maximálisan szereplő kifejezések száma,
- valamint az asszociációs hálózat korrelációs határa

A csúszkák elmozdításának hatására mind a szófelhő, mind az asszociációs hálózat azonnal frissül.

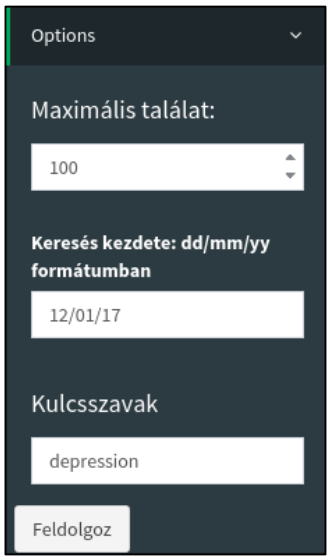


8. ábra: *Sliders menü*

Az alapértelmezett beállítások:

- minimum frekvencia: 20
- maximum kifejezésszám: 40
- korrelációs határ: 0.8

A „Wordcloud from NCBI” fülre kattintva elkezdődik az NCBI adatbázisból a feldolgozás az alapértelmezett értékek szerint. Ez lényegesen hosszabb ideig tart, mint az egy fájlból való feldolgozás .



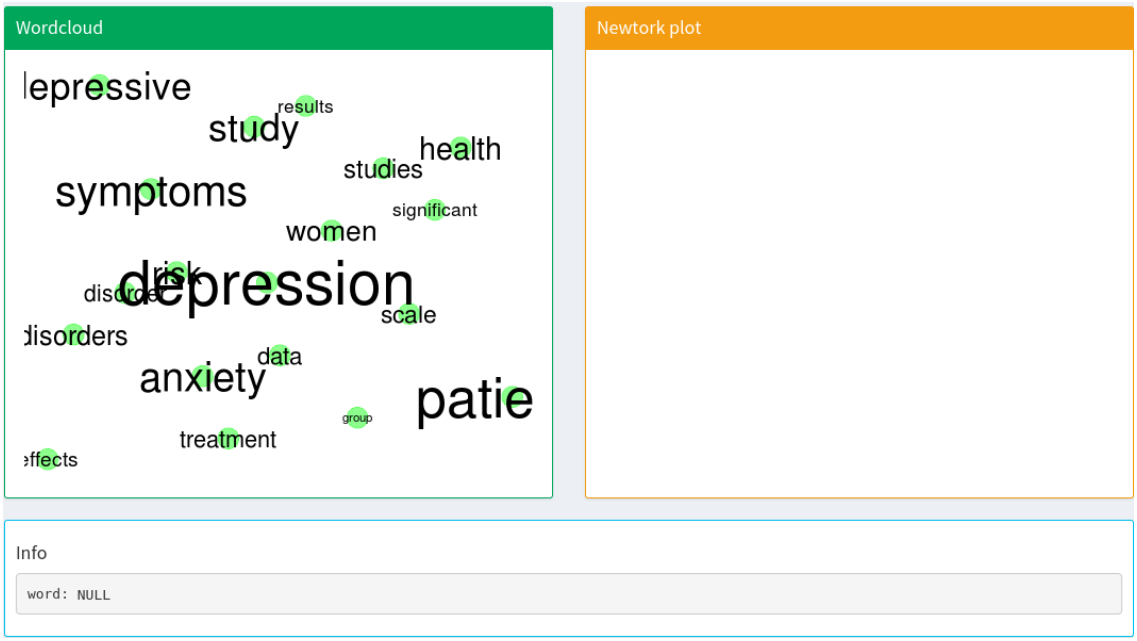
9. ábra: *Options menü*

Ezeket az alapértelmezett beállításokat, az „Options” menüben láthatjuk és állíthatjuk át nekünk megfelelően.

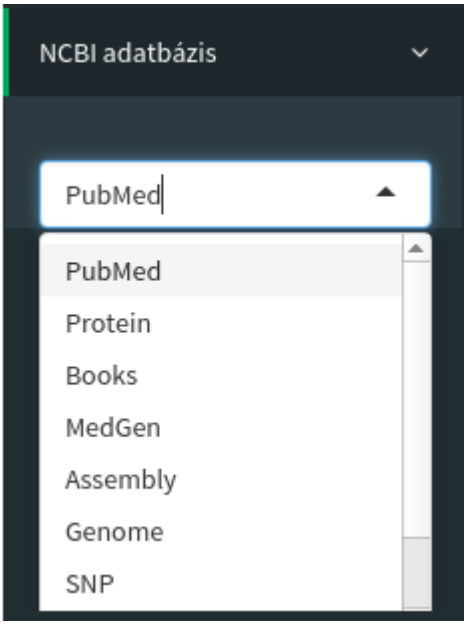
Az itt látható kép szerint a MedPub-ből az első 100 találatot vesszük át, melyeket 2017. január 12-ig visszamenőleg talál, arra a kifejezésre rákeresve, hogy „depression”.

A „Feldolgoz” feliratú gombra kattintva tudjuk megerősíteni a beállításokat, illetve azok módosításait. Ennek hatására megkezdődik a paramétereknek megfelelő adatfeldolgozás.

Az adott beállításoknak megfelelően a következő szófelhőt kapjuk:



10. ábra: *NCBI szófelhő*



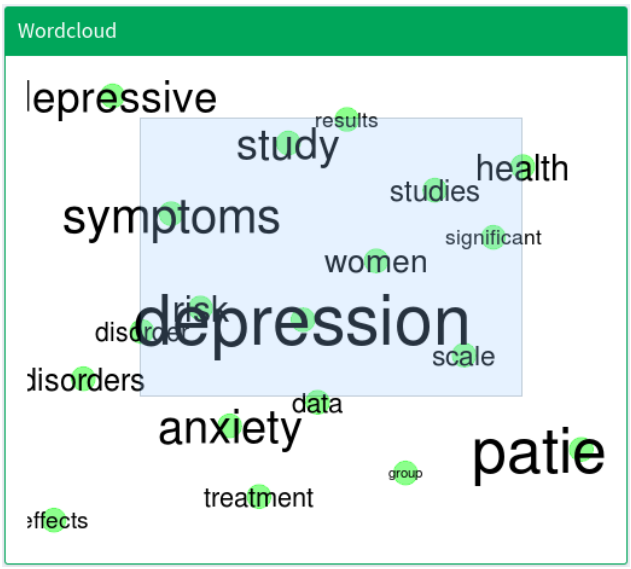
11. ábra: *NCBI menü*

Alapértelmezettként, a PubMed adatbázis van kiválasztva, amit bármikor megváltoztathatunk az „NCBI adatbázis” fülre kattintva.

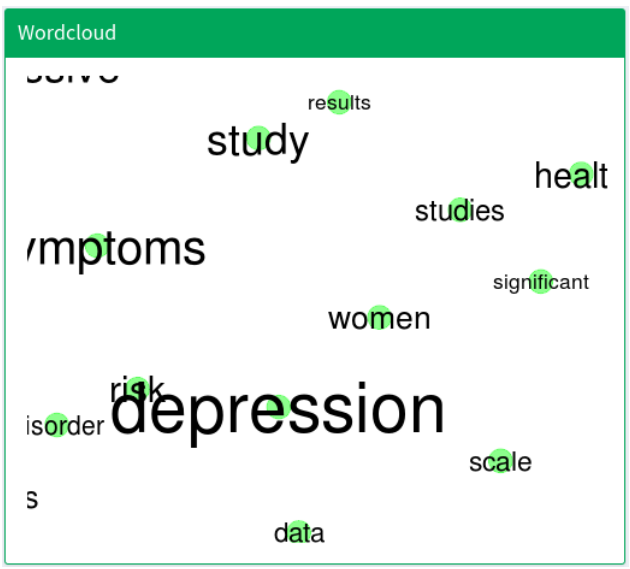
A kiválasztható adatbázisok:

- PubMed
- Protein,
- Books,
- MedGen,
- Assembly,
- SNP,
- SRA,
- Gene

A szófelhőben való nagyításhoz egyszerűen csak jeljük ki a nagyítandó területet egérhúzással, majd kattintsunk duplán a kijelölt területen belül.

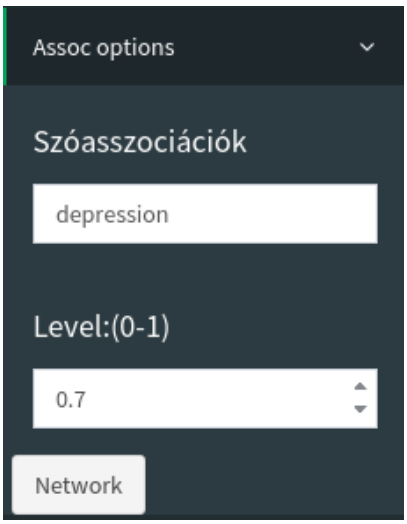


12. ábra: *Eredeti méret*



13. ábra: *Nagyított méret*

Az eredeti méret visszaállításához csak kattintsunk duplán a „Wordcloud” boxon belül.



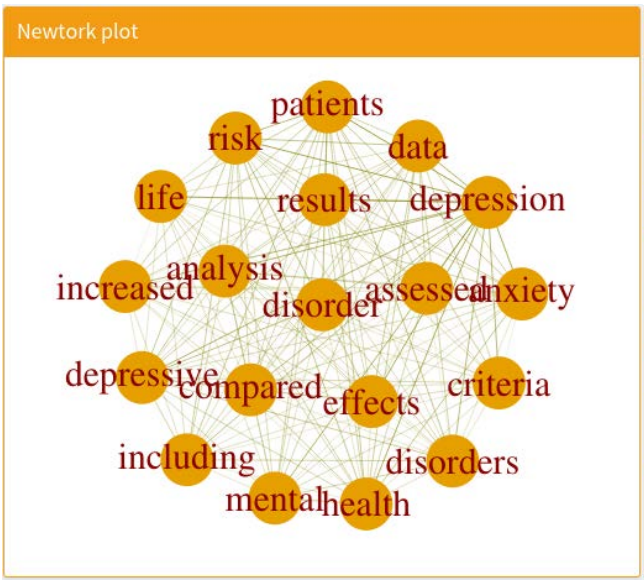
14. ábra: *Assoc menü*

Az „Assoc options” legördülő menüből állítható az asszociációs hálózat két tulajdonsága. A kívánt kezdőszóra való kattintás helyett, a „Szóasszociációk” mezőbe beírva is beállíthatjuk az asszociációs hálózat kezdő szavát, szavait.

A „Level:(0-1)” mezőben pedig kiválaszthatjuk a kívánt korrelációs határt.

Mint korábban említettem, a „Sliders” menünél is állítható a korrelációs határ. Amint időm engedi, ezen módosítani fogok, hogy ezt csak itt lehessen állítani.

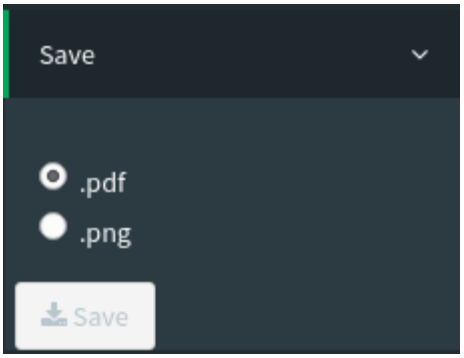
Az asszociációs hálózaton a nagyítás görgővel működik. Vigyük az egeret az ábra fölé, majd kezdjünk el görgetni. Ez megfogható az egérgombot nyomva tartva, és mozgatható is.



15. ábra: *Eredeti méret*

16. ábra: *Nagyított méret*

A „Save” fülre kattintva választhatjuk ki, milyen formátumba szeretnénk menteni a kirajzolt információkat. Alapértelmezettként a pdf formátum van kiválasztva. A kívánt formátum kiválasztása után kattintsunk a „Save” gombra, hogy a `downloadHandler` segítségével beállíthassuk a mentés útvonalát, és a menteni kívánt fájl nevét.



17. ábra*: Save menü*

A mentés még a feladat készítésének legelején készült el, mikor még nem használtam dashboardot a felhasználói felületen. A program bemutatásának készítése során derült ki, hogy így nem a korpuszok vizualizációját menti le a program, hanem magát a beállított dashboard képét. Azt az állapotot, mintha semmi nem lenne ábrázolva.

6.7. További lépések

Mind a külső, mind a belső konzulensem azt javasolta, hogy az alapértelmezett beállítások egy külső fájl segítségével történjenek meg. Jelen pillanatban, minden indításnál ugyanazon értékekre ugranak vissza ezek a beállítások. Az az elképzelés, hogy ezek után, a legelső indításnál vegye fel a most alapértelmezett

értékeket, a többinél pedig az utoljára érvényben lévő beállításokat jegyezze meg a program, és a következő indításnál ezekre térjen vissza.

Ugyanígy egy külső fájl használatát javasolták a korpusz tisztításánál használt stopszavak listájának kiválasztására. Most az angol stopszavak vannak beállítva, így viszont azok is bármikor módosíthatók lesznek.

A későbbiekben szeretném a szófelhő megjelenítését módosítani, hogy ne ennyire szellősen jelenjenek meg a címkék, egyáltalán ne legyenek átfedések és állítható mértékben legyen engedélyezve a címkék elforgatása, valamint a címke egész területén lehetséges legyen az adott kifejezés visszaadása, ne csak a középpont egy bizonyos környezetében. Szeretném befejezni az asszociációs hálózat interaktivitásának átültetését az NBCI korpuszra, valamint a feladat végén keletkezett mentési problémát kijavítani.

7. Irodalomjegyzék

BDE Research Közhasznú Nonprofit Kft. (2010. június 30).
Adatgyűjtés, elemzés, alaptechnológiák elemzése.

BDE Research Közhasznú Nonprofit Kft. (2011. március 31).
Algoritmusok leírása.

Chang, W. (2016. november 01). Package ‘shiny’.
Web Application Framework for R.
Forrás: https://cran.r-project.org/web/packages/shiny/shiny.pdf

Chang, W. (2016. szeptember 20). Package ‘shinydashboard’.
Create Dashboards with 'Shiny'.
Forrás: https://cran.r-project.org/web/packages/shinydashboard/shinydashboard.pdf

Csardi, G. (2015. június 29). Package ‘igraph’.
Network Analysis and Visualization.
Forrás: https://cran.r-project.org/web/packages/igraph/igraph.pdf

Feinerer, I. (2015. július 03). Package ‘tm’.

Text Mining Package.

Forrás: https://cran.r-project.org/web/packages/tm/tm.pdf

Gorakala, s. k. (2014. március 19). *R-bloggers*.

Forrás: Build Web applications using Shiny R:

https://www.r-bloggers.com/build-web-applications-using-shiny-r/

Hester, J. (2016. augusztus 29). Package ‘memoise’.

Memoisation of Functions.

Forrás: https://cran.r-project.org/web/packages/memoise/memoise.pdf

horvimi. (2013. május 03). *Excel Tudásbázis*.

Forrás: Dashboard alapvetés: http://excel-bazis.hu/tutorial/dashboard-alapvetes

Kovalchik, S. (2016. november 2). Package ‘RISmed’.

Download Content from NCBI Databases.

Forrás: https://cran.r-project.org/web/packages/RISmed/RISmed.pdf

RStudio. (dátum nélk.) Forrás: https://www.rstudio.com/products/rstudio/

Slowikowski, K. (2016. november 24). Package ‘ggrepel’.

Repulsive Text and Label Geoms for 'ggplot2'.

Forrás: https://cran.r-project.org/web/packages/ggrepel/ggrepel.pdf

Trattner, C., Helic, D., & Strohmaier, M. (dátum nélk.).

Tag Clouds.

Forrás: http://www.christophtrattner.info/pubs/tag_cloud.pdf

Warnes, G. R. (2016. március 30). Package ‘gplots’.

Various R Programming Tools for Plotting Data.

Forrás: https://cran.r-project.org/web/packages/gplots/gplots.pdf

Wickham, H. (2016. december 30). Package ‘ggplot2’.

Create Elegant Data Visualisations Using the Grammar of Graphics. Forrás: https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf

Wikipedia. (dátum nélk.). *Document-term matrix*.

Forrás: https://en.wikipedia.org/wiki/Document-term_matrix

Wikipedia. (dátum nélk.). *PubMed*.

Forrás: https://en.wikipedia.org/wiki/PubMed

Wikipedia. (dátum nélk.). *Qt (software)*.

Forrás: https://en.wikipedia.org/wiki/Qt_(software)

Wikipedia. (dátum nélk.). *R (programozási nyelv)*.

Forrás: https://hu.wikipedia.org/wiki/R_(programozási_nyelv)

Wikipedia. (dátum nélk.). *RStudio*.

Forrás: https://en.wikipedia.org/wiki/RStudio

Wikipedia. (dátum nélk.). *Szövegbányászat*.

Forrás: https://hu.wikipedia.org/wiki/Szövegbányászat

8. Függelék/Mellékletek

Szakdolgozatomhoz melléklek egy CD-t, melynek tartalma:

- a megírt program forráskódja a roxygen-hez használatos kommentekkel;
- a szakdolgozat szövege.

9. Nyilatkozat

Alulírott Lapatinszki András PTE-MIK hallgatója kijelentem, hogy a szakdolgozatomat nem megengedett segítség nélkül, saját magam készítettem és csak az Irodalomjegyzékben megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból vettem, egyértelműen, a forrás megadásával megjelöltem.

Pécs(2017.01.10.) .....

Lapatinszki András