Szövegbányászat és mesterséges intelligencia R-ben

Sebők Miklós, Ring Orsolya, Máté Ákos

2021-04-16 23:03:34

# Üdvözöljük!

A szövegek kvantitatív elemzése (quantitative text analysis – QTA) a nemzetközi társadalomtudomány egyik leggyorsabban fejlődő irányzata. A szövegek és más minőségi adatok (filmek, képek) elemzése annyiban különbözik a mennyiségi (kvantitatív) adatokétól, hogy nyers formájukban még nem alkalmasak arra, hogy statisztikai, illetve ökonometriai elemzés alá vessük őket, s így további módszertani problémákat vetnek fel, melyek speciális tárgyalása szükséges. A tervezett kötetben bemutatott példák többsége a politikatudományhoz kapcsolódik, de más alkalmazási területekre is kitér.

# 1 Bevezetés

## 1.1 A kötet céljai és főbb témái

A szövegek adatként való értelmezése (*text as data*) és kvantitatív elemzése (*quantitative text analysis*), avagy a szövegbányászat (*text mining*) a nemzetközi társadalomtudomány egyik leggyorsabban fejlődő irányzata. A szövegek és más kvalitatív adatok (filmek, képek) elemzése annyiban különbözik a mennyiségi (kvantitatív) adatokétól, hogy nyers formájukban még nem alkalmasak arra, hogy statisztikai, illetve ökonometriai elemzés alá vessük őket, s így további módszertani problémákat vetnek fel, melyeknek speciális tárgyalása szükséges. Jelen kötet a szövegbányászat és a mesterséges intelligencia társadalomtudományi alkalmazásának ilyen speciális problémáiba, valamint ezek gyakorlati megoldásába nyújt bevezetést. Közvetlen előzménye a Sebők Miklós által szerkesztett *Kvantitatív szövegelemzés és szövegbányászat a politikatudományban* (**Sebők, 2016**) címmel megjelent könyv, melynek az itt közölt szöveg sok szempontból folytatásaként és egyben kiegészítéseként értelmezhető. A Kvantitatív szövegelemzés kötet fókuszában elsősorban a magyar tudományos diskurzusban még kevésbé bevett alapfogalmak és eljárások bemutatása volt, s kevésbé volt gyakorlatorientált – bár számos fejezetben már ott is konkrét kódrészleteket elemeztünk. Míg az előző kötet az egyes kódolási eljárásokat, illetve ezek kutatás-módszertani előnyeit és hátrányait ismertette, itt a társadalomtudományi elemzésének során használható kvantitatív szövegelemzés legfontosabb gyakorlati feladatait vesszük sorra. Ennyiben könyvünk a magyar tankönyvpiacon elsőként ismerteti lépésről-lépésre a nemzetközi társadalomtudományban használatos kvantitatív szövegelemzési eljárásokat. Példáinkat az R programnyelven írtuk, mely a nemzetközi társadalomtudományi vizsgálatok két leggyakrabban használatos környezetének egyike (a másik a python).

## 1.2 A kötet témái és alkalmazási lehetőségei

**TARTALOMJEGYZÉK 2 BEKEZDÉS PLUSZ TOPIK MODELL IDE**

Az olvasó a két kötet együttes használatával olyan ismeretek birtokába kerül, melyek révén képes lesz alkalmazni a kvantitatív szövegelemzés és szövegbányászat legalapvetőbb eljárásait saját kutatására. Deduktív vagy induktív felfedező logikája fényében dönthet az adatelemzés módjáról, és a felkínált menüből kiválaszthatja a kutatási tervéhez legjobban illeszkedő megoldásokat. A kötetet végig kísérő konkrét példák segítségével pedig akár reprodukálhatja is ezen eljárásokat saját kutatásában. Mindezt a kötet fejezeteiben bőséggel tárgyalt R-scriptek (kis programok) részletes leírása is segíti. A kötet két fő célcsoportjaként így a társadalomtudományi kutatói és felsőoktatási hallgatói-oktatói közösséget határozhatjuk meg. Az oktatási alkalmazást segítheti a fontosabb fogalmak magyar és angol nyelvű szószedete, valamint több helyen a további olvasásra ajánlott szakirodalom szerepeltetése. A kötet honlapján (www.hunminer.hu) közvetlenül is elérhetőek a felhasznált adatbázisok és kódok.

## 1.3 Használati utasítás

A könyv célja hogy egy keresztmetszeti képet adjon a szövegbányászat R programnyelven használatos eszközeiről. A fejezetekben ezért a magyarázó szövegben maga az R kód is megtalálható, illetve a lefuttatott kód eredményei is láthatóak. A lenti példában a sötét háttér az R környezetet jelöli, ahol az R kód betűtípusa is eltérő a főszövegtől. A kód eredménye pedig a #> kezdetű sorokban van, ezzel szimulálva az R console ablakát.

# példa R kód  
  
1 + 1

## [1] 2

Az egyes fejezetekben szereplő kódrészleteket egymás után sorrendben bemásolva és lefuttatva a saját R környezetünkbe tudjuk reprodukálni a könyvben szereplő technikákat. A [Függelékben](#fuggelek) részletesebben is foglalkozunk az R és RStudio beállításaival, használatával. Az ajánlott R minimum verzió, az 4.0.0, illetve az ajánlott minimum RStudio verzió az 1.4.0000.[[1]](#footnote-23)

A könyvhöz tartozik egy HunMineR nevű R csomag is, ami tartalmazza az összes adatbázist amit a fejezetkben használunk, így az adatbeviteli problémákat elkerülve lehet gyakorolni a szövegbányászatot. A könyv megjelenésének idejében a csomag még nem került be a központi R CRAN csomag repozitóriumba, hanem a poltextLAB GitHub repozitóriumából tölthető le.

### 1.3.1 HunMineR telepítése és használata

A Windows rendszert használóknak először az installr csomagot kell telepíteni, majd annak segítségével letölteni az Rtools nevű programot (az OS X és Linux rendszerek esetében erre a lépésre nincs szükség). A lenti kód futtatásával ezek a lépések autómatikusan megtörténnek.

# az installr csomag letöltése és installálása  
install.packages("installr")  
  
# az Rtools.exe fájl letöltése és installálása  
installr::install.Rtools()

Ezt követően a devtools csomagban található install\_github paranccsal tudjuk telepíteni a HunMineR csomagot, a lenti kód lefuttatásával.

# A devtools csomag letöltése és installálása  
install.packages("devtools")  
  
# A HunMineR csomag letöltése és installálása  
devtools::install\_github("poltextlab/HunMineR")

Ezt követően a data függvénnyel tudjuk megnézni hogy pontosan milyen adatbazisok szerepelnek a csomagban, illetve az egyes adatbázisok részletes leírását is meg tudjuk nézni. Hogyha egy adatbázisról szeretnénk többet megtudni, akkor a kiegészítő információkat ?adatbazis\_neve megoldással tudjuk megnézni.[[2]](#footnote-26)

# A HunMineR csomag betöltése  
library(HunMineR)  
  
# csomagban lévő adatok listázása  
data(package = "HunMineR")  
  
# A miniszterelnöki beszédek minta adatbázisának részletei  
?data\_miniszterelnokok

## 1.4 Köszönetnyilvánítás

A jelen kötet az ELKH Társadalomtudományi Kutatóközpontban, a poltextLAB szövegbányászati kutatócsoport (<http://poltextlab.com/>) keretében készült. A kötet fejezetei – két kivétellel – Sebők Miklós, Máté Ákos és Ring Orsolya közös munkájának eredményei. Az *Alapfogalmak*, illetve a *Szövegösszehasonlítás* fejezetben társszerzőként Székely Anna, az *Osztályozás* fejezetben Kacsuk Zoltán is közreműködött. A kézirat a szerzők több éves oktatási gyakorlatára, a hallgatóktól kapott visszajelzésekre építve készült el. Köszönjük a Bibó Szakkollégiumban (2021), a Rajk Szakkollégiumban (2019-2021), valamint a Széchenyi Szakkollégiumban (2019) tartott féléves, valamint a Corvinus Egyetemen és a Társadalomtudományi Kutatóközpontban tartott rövidebb képzési alkalmak résztvevőinek visszajelzéseit. Köszönjük a projekt gyakornokainak, Czene-Joó Máténak, Kaló Eszternek, Meleg Andrásnak, Lovász Dorottyának, Nagy Orsolyának, valamint kutatász asszisztenseinek, Balázs Gergőnek és Gelányi Péternek a segítségét a kézirat végleges formába öntése során. Hálával tartozunk továbbá a Társadalomtudományi Kutatóközpont Comparative Agendas Project (<https://cap.tk.hu/hu>) kutatócsoportja tagjainak, így kiemelten Boda Zsoltnak, Molnár Csabának és Pokornyi Zsanettnek a kötetben használt korpuszok sok éves előkészítéséért. Köszönettel tartozunk a több fejezet alapjául szolgáló elemzéseink és publikációink társszerzőinek, Barczikay Tamásnak, Berki Tamásnak, Kubik Bálintnak, Molnár Csabának és Szabó Martina Katalinnak. Köszönjük **SZAKMAI LEKTOR** hasznos megjegyzéseit, Fedinec Csilla olvasószerkesztő alapos munkáját, valamint a Typotex Kiadó rugalmasságát és színvonalas közreműködését a könyv kiadásában! Végül, de nem utolsósorban hálásak vagyunk a kötet megvalósulásához támogatást nyújtó szervezeteknek és ösztöndíjaknak: az MTA Könyvkiadási Alapjának, a Társadalomtudományi Kutatóközpont Könyvtámogatási Alapjának, a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatalnak (NKFIH FK 123907, NKFIH FK 129018), az MTA Bolyai Ösztöndíjának. A kötet alapjául szolgáló kutatást, amelyet a Társadalomtudományi Kutatóközpont valósított meg, az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta a Mesterséges Intelligencia Nemzeti Laboratórium keretében.

# 2 Alapfogalmak

## 2.1 Elméleti alapok

A szövegek géppel való feldolgozásának és elemzésének módszertanának számos megnevezése létezik. A szövegelemzés, kvantitatív szövegelemzés, szövegbányászat, természetes nyelvfeldolgozás, automatizált szövegelemzés, automatizált tartalomelemzés, és hasonló fogalmak között nincs éles tartalami különbség. Ezek a kifejezések jellemzően ugyanarra az általánosabb kutatási irányra reflektálnak, csupán hangsúlybeli eltolódások vannak köztük, így gyakran szinonimaként is használják őket. A szövegek gépi feldolgozásával foglalkozó tudományág a Big Data forradalom részeként kezdett kialakulni, melyet az adatok egyre nagyobb és diverzebb tömegének elérhető és összegyűjthető jellege hívott életre. Ennek megfelelően az adattudomány számos különböző adatforrás – így képek, videók, hanganyagok, internetes keresési adatok, telefonok lokációs adatai és megannyi különböző információ feldolgozásával foglalkozik. A szöveg is egy, az adatbányászat érdeklődési körébe eső számos adattípus közül, melynek elemzésére egy külön kutatási irány alakult ki.

Mivel napjainkban minden másodpercben óriási mennyiségű szöveg keletkezik és válik hozzáférhetővé az interneten, egyre nagyobb az igény az ilyen jellegű források, és az emberi nyelv automatizált feldolgozására. Ebből adódóan az elemzési eszköztár is egyre szélesebb körű és szofisztikáltabb, így a tartalomelemzési és szövegbányászati ismeretekkel bíró elemzők számára rengeteg értékes információ kinyerhető. Ezt gyakran hasznosítják üzleti célokra, de a társadalomtudósok számára is rengeteg izgalmas kutatási irányt kínál a szövegbányászat eszköztára. Gondoljunk például az online sajtótermékekre, az ezekhez kapcsolódó kommentekre, vagy a politikusok beszédéire. Ezek mind-mind hatalmas mennyiségben rendelkezésre állnak, hasznosításukhoz azonban képesnek kell lenni ezeket a szövegeket összegyűjteni, a megfelelő módon feldolgozni és kiértékelni. A könyv további fejezetei ebben nyújtanak segítséget az olvasónak. Mielőtt azonban az adatkezelés és az elemzés részleteire rátérnénk, érdemes végig venni néhány elvi megfontolást, mely segítheti a leendő elemzőt az etikus, érvényes, és eredményes szövegbányászati kutatások kivitelezésében.

A nagy mennyiségben rendelkezésre álló szöveges források kiváló kutatási terepet kínálnak a társadalomtudósok számára megannyi vizsgálati kérdéshez, azonban fontos tisztában lenni vele, hogy a mindenki által elérhető adatokat is meglehetősen körültekintően, etikai szempontok figyelembevételével kell használni. Egy másik szempont, amelyet érdemes szem előtt tartani mielőtt az ember fejest ugrana az adatok végtelenjébe a 3V elve: volume, velocity, variety vagyis az adatok mérete, a keletkezésük sebessége és azok változatossága ([Brady 2019](#ref-bradyChallengeBigData2019)). Ezek mind olyan tulajdonságok, amelyek jelentősen más (és sok esetben több vagy nagyobb) kihívásokok elé állítják az adatelemzőt munkája során, mint egy hagyományos statisztikai elemzés esetén. A szövegbányászati módszerek abban is eltérnek a hagyományos társadalomtudományi elemzésektől, hogy – az adattudományokba visszanyúló gyökerei miatt – jelentős teret nyit az induktív (empiricista) kutatások számára a deduktív szemlélettel szemben. A deduktív kutatásmódszertani megközelítés esetén a kutató előre meghatározza az alkalmazandó fogalomrendszert, és azokat az elvárásokat, amelyek teljesülése esetén sikeresnek tekinti az elemzést. Az adattudományban az ilyen megközelítés a felügyelt tanulási feladatokat jellemzi, vagyis azokat a feladatokat, ahol ismert az elvárt eredmény.Ilyen például egy osztályozási feladat, amikor újságcikkeket szeretnénk különböző témakörökbe besorolni. Ebben az esetben az adatok egy részét általában kézzel kategorizáljuk, és a gépi eljárás sikerességet ehhez viszonyítjuk. Mivel az ideális eredmény (osztálycímke) ismert, a gépi teljesítmény könnyen mérhető (például a pontosságot mérve, a gép által sikeresen kategorizált cikkek százalékában kifejezve).

Az induktív megoldás esetén kevésbé egyértelmű a gépi eljárás teljesítményének mérése, hiszen ebben az esteben a rejtett mintázatok feltárást várjuk az algoritmustól, emiatt nincsenek előre meghatározott eredmények sem, amelyekhez viszonyíthatjuk a teljesítményt. Az adattudományban az ilyen feladatokat hívják felügyelet nélküli tanulásnak. Ide tartozik a klaszterelemzés, vagy a topic modellezés, melynek esetén a kutató csak azt határozza meg, hány klasztert, hány témát szeretne kinyerni, a gép pedig létrehozza az egymáshoz leghasonlóbb csoportokat. Értelemszerűen itt a kutatói validálás jóval nagyobb hangsúlyt kap, mint a deduktív megközelítés esetén.

Egy harmadik, középutas megoldás a megalapozott elmélet megközelítése, mely ötvözi az induktív és deduktív módszer előnyeit. Ennek során a kutató kidolgoz egy laza elméleti keretet, melynek alapján elvégzi az elemzést, majd az eredményeket figyelembe véve finomít a fogalmi keretén, és újabb elemzést futtat, ezt az iterációt addig folytatva, amíg a kutatás eredményeit kielégítőnek nem találja.

A szövegbányászati elemzéseket kategorizálhatjuk továbbá a gépi hozzájárulás mértéke szerint. Ennek megfelelően megkülönböztethetünk kézi, géppel támogatott és gépi eljárásokat. Mindhárom megközelítésnek megvan a maga előnye, a kézi megoldások esetén valószínűbb, hogy azt mérjük a szövegünkben amit mérni szeretnénk (például bizonyos szakpolitikai tartalmat), ugyanakkor idő és költségigényes. A gépi eljárások ezzel szemben költséghatékonyak és gyorsak, de fennáll a veszélye, hogy nem azt mérjük, amit eredetileg mérni szerettünk volna (ennek megállapításában ismét a validálás kap kulcsszerepet). Továbbá lehetséges kézzel támogatott gépi megoldások alkalmazása, ahol a humán és a gépi elemzés ideális arányának megtalálása jelenti a fő kihívást.

## 2.2 Fogalmi alapok

Miután áttekintettük a szövegbányászatban használatos elméleti megközelítéseket, érdemes tisztázni a fogalmi alapokat is. A szövegbányászat szempontjából a szöveg is egy adat, az elemzéshez használatos strukturált adathalmazt pedig korpusznak nevezzük. A korpusz az összes szövegünket jelöli, ennek részegységei a dokumentumok. Ha például a Magyar Nemzet cikkeit kívánjuk elemezni, a kiválasztott időszak összes cikke lesz a teljes korpuszunk, az egyes cikkek pedig a dokumentumaink. Az elemzés mindig egy meghatározott (téma-)területre („domain"-re) koncentrál. E (téma-)terület (domain) utalhat a nyelvre, amelyen a szövegek íródtak, vagy a specifikus tartalomra, amelyet vizsgálunk, de mindenképpen meghatározza a szöveg szókészletével kapcsolatos várakozásainkat. Más lesz tehát a szóhasználat egy bulvárlap cikkeiben mint egy tudományos szaklap cikkeiben, melynek elsősorban akkor van jelentősége, ha szótár alapú elemzéseket készítünk. A szótár alapú elemzések során olyan szószedetket hozunk létre, amelyek segíthetnek a kutatásunk szempontjából érdekes témák vagy tartalmak azonosításában. Így például létrehozhatunk pozitív és negatív szótárakat, vagy a gazdasági és külpolitikai témákhoz kapcsolódó szótárakat, melyek segíthetnek azonosítani, hogy adott dokumentum inkább gazdasági vagy inkább külpolitikai témákat tárgyal. Léteznek előre elkészített szótárak (angol nyelven például a Bing Liu által fejlesztett szótár egy jól ismert és széles körben alkalmazható példa ([Liu 2010](#ref-liuSentimentAnalysisSubjectivity2010))), azonban fontos fejben tartani, hogy a vizsgált téma specifikus nyelvezete jellemzően meghatározza azt, hogy egy-egy szótárba milyen kifejezéseknek kellene kerülniük.

Már említésre került, hogy egy szövegbányászati elemzés során a szöveg is adatként kezelendő. Tehát hasonló módon gondolhatunk az elemzendő szövegeinkre, mint egy statisztikai elemzésre szánt adatbázisra, annak csupán, reprezentációja tér el az utóbbitól. Tehát míg egy statisztikai elemzésre szánt táblázatban elsősorban számokat és adott esetben kategorikus változókat reprezentáló karakterláncokat (sztringeket) pl. „férfi“/”nő“, „falu”/“város,” találunk, addig a szöveges adatokban első ránézésére nem tűnik ki gépileg értelmezhető struktúra. Ahhoz, hogy a szövegeink a gépi elemzés számára feldolgozhatóvá váljanak, annak reprezentációját kell megváltoztatni, vagyis strukturálatlan adathalmazból strukturált adathalmazt kell létrehozni, melyet jellemzően a szövegek mátrixszá alakításával tesszünk meg. A mátrixszá alakítás első hallásra bonyolult eljárás benyomását keltheti, azonban a gyakorlatban egy meglehetősen egyszerű transzformációról van szó, melynek eredményeként a szavakat számokkal reprezentáljuk. A könnyebb megértés érdekében vegyük az alábbi példát: tekintsük a három példamondatot a három elemzendő dokumentumnak, ezek összességét pedig a korpuszunknak.

*1. Az Európai Unió 27 tagországának egyike Magyarország.*

*2. Magyarország 2004-ben csatlakozott az Európai Unóhoz.*

*3. Szlovákia, akárcsak Magyarország, 2004-ben lett ez Európai Unió tagja.*

A példamondatok dokumentum-kifejezés mátrixsza az alábbi táblázat szerint fog kinézni.[[3]](#footnote-35)

(#tab:unnamed-chunk-8)Dokumentum-kifejezés mátrix három példamondattal

Dokumentum száma

27

2004-ben

akárcsak

az

csatlakozott

egyike

Európai

lett

Magyarország

Szlovákia

tagja

tagjának

Unió

Unióhoz

1

1

0

0

1

0

1

1

0

1

0

0

1

1

0

2

0

1

0

1

1

0

1

0

1

0

0

0

0

1

3

0

1

1

1

0

0

1

1

1

1

1

0

1

0

A dokumentum-kifejezés mátrixban minden dokumentumot egy vektor (értsd: egy sor) reprezentál, az eltérő kifejezések pedig külön oszlopokat kapnak. Tehát a fenti példában minden dokumentumunk egy 14 elemű vektorként jelenik meg, melynek elemei azt jelölik, hogy milyen gyakran szerepel az adott kifejezés a dokumentumban. A dokumentum-kifejezés mátrixok egy jellemző tulajdonsága, hogy igen nagy dimenziókkal rendelkezhetnek (értsd: sok sorral és sok oszloppal), hiszen minden kifejezést külön oszlopként reprezentálnak. Egy sok dokumentumból álló vagy egy témák tekintetében változatos korpusz esetén a kifejezés mátrix elemeinek jelentős része 0 lesz, hiszen számos olyan kifejezés lesz az egyes dokumentumokban, amely más dokumentumban nem szerepel. A sok nullát tartalmazó mátrixot hívjuk ritka mátrixnak. Az adatok jobb kezelhetőségének érdekben a ritka mátrixot valamilyen dimenzióredukciós eljárással sűrű mátrixszá lehet alakítani (például a nagyon ritka kifejezések eltávolításával, vagy valamilyen súlyozáson alapuló eljárással).

## 2.3 A szövegbányászat alapelvei

Végezetül pedig a módszertani fogalmak tisztázást követően néhány elméleti megfontolást osztanánk meg a Grimmer és Steward ([2013](#ref-grimmer2013text)) által megfogalmazott alapelvek nyomán, melyek hasznos útravalóul szolgálhatnak a szövegbányászattal ismerkedő kutatók számára.

*1. A szövegbányászat rossz, de hasznos*

Az emberi nyelv egy meglehetősen bonyolult rendszer, így egy szöveg jelentésének, érzelmi telítettségének értelmezése meglehetősen eltérő lehet különböző olvasók által, így nem meglepő, hogy egy gép sok esetben csak korlátozott eredményeket képes felmutatni ezen feladatok teljesítésében. Ettől függetlenül nem elvitatható a szövegbányászati modellek hasznossága, hiszen olyan mennyiségű szöveg válik feldolgozhatóvá, amely gépi támogatás nélkül elképzelhetetlen volna, mindemellett azonban fejben kell tartani a módszertan korlátait is.

*2. A kvantitatív modellek kiegészítik az embert, nem helyettesítik azt*

A kvantitatív eszközökkel történő elemzés nem szünteti meg a szövegek elolvasásának szükségességét, hiszen egészen más információk kinyerését teszi lehetővé egy kvantitatív megközelítéssel szemben. Emiatt a kvantitatív szövegelemzés talán legfontosabb kihívása, hogy a kutató megtalálja a gépi és a humán erőforrások együttes hasznosításának legjobb módját.

*3. Nincs legjobb modell*

Minden kutatáshoz meg kell találni a leginkább alkalmas modellt a kutatási kérdés, a rendelkezésre álló adatok, és a kutatói szándék alapján. Gyakran különböző eljárások kombinálása vezethet egy specifikus probléma legjobb megoldására. Azonban minden esetben az eredmények értékelésére kell támaszkodni, hogy megállapíthassuk egy modell teljesítményét adott problémára és szövegkorpuszra nézve.

*4. Validálás, validálás, validálás! [érvényesítés?]*

Mivel az automatizált szövegelemzés számos esetben jelentősen lecsökkenti az elemzéshez szükséges időt és energiát, csábító lehet a gondolat, hogy ezekhez a módszerekhez forduljon a kutató, ugyanakkor nem szabad elfelejteni, hogy az elemzés csupán a kezdeti lépés, hiszen a kutatónak érvényesítenie (validálnia) kell az eredményeket ahhoz, hogy valóban megbízható következtetésekere jussunk. Az érvényesítés lényege egy felügyelet nélküli modell esetén (ahol az elvárt eredmények nem ismertek, így a teljesítmény nem tesztelhető), hogy meggyőződjünk arról, hogy egy felügyelt modellel (olyan modellel, ahol az elvárt eredmény ismert, így ellenőrizhető) egyenértékű eredményeket hozzon. Ezen elvárás teljesítése gyakran nem egyszerű, azonban az eljárások alapos kiértékelést (validálást) nélkülöző alkalmazása meglehetősen kétesélyes eredményekhez vezethet, emiatt érdemes megfelelő alapossággal eljárni az érvényesítés során.

# 3 Az adatkezelés R-ben

## 3.1 Adatok importálása és exportálása

library(readr)  
library(dplyr)  
library(gapminder)  
library(stringr)

Az adatok importálására az R alapfüggvénye mellett több csomag is megoldást kínál. Ezek közül a könyv írásakor a legnépszerűbbek a readr és a rio csomagok. A szövegek különböző karakterkódolásának problémáját tapasztalataink szerint a legjobban a readr csomag read\_csv() függvénye kezeli, ezért legtöbbször ezt fogjuk használni a .csv állományok beolvasására. Amennyiben kihasználjuk az RStudio projekt opcióját (lásd a [Függelékben](#projektmunka)) akkor elegendő csak az elérni kívánt adatok relativ elérési útját megadni (*relative path*). Ideális esetben az adataink egy csv fájlban vannak (*comma separated values*), ahol az egyes értékeket vesszők (vagy egyéb speciális karakterek) választják el. Ez esetben a read\_delim() függvényt használjuk. A beolvasásnál egyből el is tároljuk az adatokat egy objektumban. A sep = opcióval tudjuk a szeparátor karaktert beállítani, mert előfordulhat hogy vessző helyett pontosvessző tagolja az adatainkat.

df <- read\_csv("data/adatfile.csv")

Az R képes linkről letölteni fájlokat, elég megadnunk egy működő elérési útvonalat.

df\_online <- read.csv("https://www.qta.tk.mta.hu/adatok/adatfile.csv")

Az R package ökoszisztémája kellően változatos ahhoz, hogy gyakorlatilag bármilyen inputtal meg tudjon birkózni. Az Excel fájlokat a readxl csomagot használva tudjuk betölteni (a csomagok installálásával kapcsolatban lásd a [Függeléket](#packages)), a read\_excel()-t használva. A leggyakoribb statisztikai programok formátumait pedig a haven csomag tudja kezelni (például Stata, Spss, SAS). A szintaxis itt is hasonló: read\_stata, read\_spss, read\_sas.

### 3.1.1 Szöveges dokumentumok importálása

A nagy mennyiségű szöveges dokumentum (a legyakrabban előforduló kiterjesztések: .txt, .doc, .pdf, .json, .csv, .xml, .rtf, .odt) betöltésére a legalkalmasabb a readtext package. Az alábbi példa azt mutatja be, hogy hogyan tudjuk beolvasni egy adott mappából az összes .txt kiterjesztésű fájlt, anélkül hogy egyenként kellene megadnunk a fájlok neveit. A kódsorban szereplő \* karakter azt ebben a környezetben jelenti, hogy bármilyen fájl, ami .txt-re végződik. Amennyiben a fájlok nevei tartalmaznak valamilyen metaadatot, akkor ezt is be tudjuk olvasni a betöltés során. Ilyen meta adat lehet például egy parlamenti felszólalásnál a felszólaló neve és a beszéd ideje és párttagsága (például: kovacsjanos\_1994\_fkgp.txt).

df\_text <- readtext(  
 "data/\*.txt",  
 docvarsfrom = "filenames",  
 dvsep = "\_",  
 docvarnames = c("nev", "ev", "part")  
)

## 3.2 Adatok exportálása

Az adatainkat R-ből a write.csv()-vel exportálhatjuk a kívánt helyre, .csv formátumban. Az openxlsx csomaggal .xls és .xlsx Excel formátumokba is tudunk exportálni. Az R rendelkezik saját, .Rds és .Rda kiterjesztésű, tömörített fájlformátummal.[[4]](#footnote-41) Mivel ezeket csak az R-ben nyithatjuk meg, érdemes a köztes, hosszadalmas számítást igénylő lépések elmentésére használni, a saveRDS() és a save() parancsokkal.

## 3.3 A pipe operátor

Az úgynevezett **pipe** operátor alapjaiban határozta meg a modern R fejlődését és a népszerű package ökoszisztéma, a **tidyverse**, egyik alapköve. Úgy gondoljuk, hogy a **tidyverse** és a **pipe** egyszerűbbé teszi elsajátítani az R használatát, ezért mi is erre helyezzük a hangsúlyt.[[5]](#footnote-43) Vizuálisan a pipe operátor így néz ki: %>% és arra szolgál hogy a kódban több egymáshoz kapcsolódó műveletet egybefűzzünk.[[6]](#footnote-45) Technikailag a pipe a bal oldali elemet adja meg a jobb oldali függvény első argumentumának. A lenti példa ugyanazt a folyamatot írja le az alap R (*base R*), illetve a pipe használatával.[[7]](#footnote-46) Miközben a kódot olvassuk érdemes a pipe-ot “*és aztán*”-nak fordítani.

reggeli(oltozkodes(felkeles(ebredes(en, idopont = "8:00"), oldal = "jobb"), nadrag = TRUE,   
 ing = TRUE))  
  
en %>% ebredes(idopont = "8:00") %>% felkeles(oldal = "jobb") %>% oltozkodes(nadrag = TRUE,   
 ing = TRUE) %>% reggeli()

A fenti példa is jól mutatja, hogy a pipe a bal oldali elemet fogja a jobb oldali függvény első elemének berakni. A fejezet további részeiben még bőven fogunk gyakorlati példát találni a **pipe** használatára. Mivel az itt bemutatott példák az alkalmazásoknak csak egy relatíve szűk körét mutatják be, érdemes átolvasni a csomagokhoz tartozó dokumentációt, illetve ha van, akkor tanulmányozni a működést demonstráló bemutató oldalakat is.

## 3.4 Műveletek a data framekkel

A **data frame** az egyik leghasznosabb és leggyakrabban használt adat tárolási mód az R-ben (a részletesebb leírás a [Függelékben](#data-frame) található). Ebben az alfejezetben azt mutatjuk be a dplyr és gapminder csomagok segítségével, hogy hogyan lehet hatékonyan dolgozni vele. A dplyr az egyik legnépszerűbb R csomag, a **tidyverse** része. A gapminder csomag pedig a példa adatbázisunkat tartalmazza, amiben a világ országainak különböző gazdasági és társadalmi mutatói vannak.

### 3.4.1 Megfigyelések szűrése: filter()

A sorok (megfigyelések) szűréséhez a dplyr csomag filter() parancsát használva lehetőségünk van arra hogy egy vagy több kritérium alapján szűkítsük az adatbázisunkat. A lenti példában azokat megfigyeléseket tartjuk meg, ahol az év 1962 és a várható élettartam nagyobb mint 72 év.

gapminder %>%  
 filter(year == 1962, lifeExp > 72)  
#> # A tibble: 5 x 6  
#> country continent year lifeExp pop gdpPercap  
#> <fct> <fct> <int> <dbl> <int> <dbl>  
#> 1 Denmark Europe 1962 72.4 4646899 13583.  
#> 2 Iceland Europe 1962 73.7 182053 10350.  
#> 3 Netherlands Europe 1962 73.2 11805689 12791.  
#> 4 Norway Europe 1962 73.5 3638919 13450.  
#> 5 Sweden Europe 1962 73.4 7561588 12329.

De ugyanígy leválogathatjuk a data frame-ből az adatokat akkor is hogyha egy karakter változó alapján szeretnénk szűrni.

gapminder %>%  
 filter(country == "Sweden", year > 1990)  
#> # A tibble: 4 x 6  
#> country continent year lifeExp pop gdpPercap  
#> <fct> <fct> <int> <dbl> <int> <dbl>  
#> 1 Sweden Europe 1992 78.2 8718867 23880.  
#> 2 Sweden Europe 1997 79.4 8897619 25267.  
#> 3 Sweden Europe 2002 80.0 8954175 29342.  
#> 4 Sweden Europe 2007 80.9 9031088 33860.

Itt tehát a data frame azon sorait szeretnénk látni, ahol az ország megegyezik a „Sweden" karakterlánccal az év pedig nagyobb, mint 1990.

### 3.4.2 Változók kiválogatása: select()

A select() függvény segítségével válogathatunk oszlopokat a data frame-ből. A változók kiválasztására több megoldás is van. A dplyr csomag tartalmaz apróbb kisegítő függvényeket, amik megkönnyítik a nagy adatbázisok esetén a változók kiválogatását a nevük alapján. Ezek a függvények a contains(), starts\_with(), ends\_with(), matches() és beszédesen arra szolgálnak hogy bizonyos nevű változókat ne kelljen egyenként felsorolni. A select()-en belüli változó sorrend egyben az eredmény data frame változó sorrendjét is megadja. A negatív kiválasztás is lehetséges, ebben az esetben egy - kell tennünk a nem kívánt változó(k) elé (pl.: select(df, year, country, -continent).

gapminder %>%  
 select(contains("ea"), starts\_with("co"), pop)  
#> # A tibble: 1,704 x 4  
#> year country continent pop  
#> <int> <fct> <fct> <int>  
#> 1 1952 Afghanistan Asia 8425333  
#> 2 1957 Afghanistan Asia 9240934  
#> 3 1962 Afghanistan Asia 10267083  
#> 4 1967 Afghanistan Asia 11537966  
#> 5 1972 Afghanistan Asia 13079460  
#> 6 1977 Afghanistan Asia 14880372  
#> 7 1982 Afghanistan Asia 12881816  
#> 8 1987 Afghanistan Asia 13867957  
#> 9 1992 Afghanistan Asia 16317921  
#> 10 1997 Afghanistan Asia 22227415  
#> # ... with 1,694 more rows

Az így kiválogatott változókból létrehozhatunk és objektumként eltárolhatunk egy új **data framet**, amivek azután tovább dolgozhatunk, vagy kiírathatjuk például .csv fájlba, vagy elmenthetjük a saveRDS segítségével.

gapminder\_select <- gapminder %>%  
 select(contains("ea"), starts\_with("co"), pop)

write.csv(gapminder\_select, "gapminder\_select.csv")

saveRDS(gapminder\_select, "gapminder\_select.Rds")

A saveRDS segítségével elmentett fájlt később a readRDS() függvénnyel olvashatjuk be, majd onnan folytassuk a munkát, ahol korábban abbahagytuk.

readRDS("gapminder\_select.Rds")

### 3.4.3 Új változók létrehozása: mutate()

Az elemzési munkafolyamat elkerülhetetlen része hogy új változókat hozzunk létre, vagy a meglévőket módosítsuk. Ezt a mutate()-el tehetjük meg, ahol a szintaxis a következő: mutate(data frame, uj valtozo = ertekek). Példaként kiszámoljuk a Svéd GDP-t (milliárd dollárban) 1992-től kezdve. A mutate() alkalmazásával részletesebben is foglalkozunk a szövegek előkészítésével foglalkozó fejezetben.

gapminder %>%  
 filter(country == "Sweden", year >= 1992) %>%  
 mutate(gdp = (gdpPercap \* pop) / 10^9)  
#> # A tibble: 4 x 7  
#> country continent year lifeExp pop gdpPercap gdp  
#> <fct> <fct> <int> <dbl> <int> <dbl> <dbl>  
#> 1 Sweden Europe 1992 78.2 8718867 23880. 208.  
#> 2 Sweden Europe 1997 79.4 8897619 25267. 225.  
#> 3 Sweden Europe 2002 80.0 8954175 29342. 263.  
#> 4 Sweden Europe 2007 80.9 9031088 33860. 306.

### 3.4.4 Csoportonkénti statisztikák: group\_by() és summarize()

Az adataink részletesebb és alaposabb megismerésében segítenek a különböző szintű leíró statisztikai adatok. A szintek megadására a group\_by() használható, a csoportokon belüli számításokhoz pedig a summarize(). A lenti példa azt illusztrálja, hogyha kontinensenként csoportosítjuk a gapminder data framet, akkor a summarise() használatával megkaphatjuk a megfigyelések számát, illetve az átlagos per capita GDP-t. A summarise() a mutate() közeli rokona, hasonló szintaxissal és logikával használható. Ezt a függvény párost fogjuk majd használni a szöveges adataink leíró statisztikáinál is a 4. fejezetben.

gapminder %>%  
 group\_by(continent) %>%  
 summarise(megfigyelesek = n(), atlag\_gdp = mean(gdpPercap))  
#> # A tibble: 5 x 3  
#> continent megfigyelesek atlag\_gdp  
#> <fct> <int> <dbl>  
#> 1 Africa 624 2194.  
#> 2 Americas 300 7136.  
#> 3 Asia 396 7902.  
#> 4 Europe 360 14469.  
#> 5 Oceania 24 18622.

## 3.5 Munka karakter vektorokkal[[8]](#footnote-53)

A szöveges adatokkal (karakter stringekkel) való munka elkerülhetetlen velejárója hogy a felesleges szövegelemeket, karaktereket el kell távolítanunk ahhoz hogy az elemzésünk hatásfoka javuljon (erről részletesebben a 3. fejezetben lesz szó). Erre a célra a stringr csomagot fogjuk használni, kombinálva a korábban bemutatott mutate()-el. A stringr függvények az str\_ előtaggal kezdődnek és eléggé beszédes nevekkel rendelkeznek. Egy gyakran előforduló probléma, hogy extra szóközök maradnak a szövegben, vagy bizonyos szavakról, karakterkombinációkról tudjuk hogy nem kellenek az elemzésünkhöz. Ebben az esetben egy vagy több *regular expression* (**regex**) használatával tudjuk pontosan kijelölni hogy a karakter sornak melyik részét akarjuk módosítani. A legegyszerűbb formája a regexeknek, hogyha pontosan tudjuk milyen szöveget akarunk megtalálni. A kísérletezésre az str\_view()-t használjuk, ami megjeleníti hogy a megadott regex mintánk pontosan mit jelöl.

szoveg <- c("gitar", "ukulele", "nagybogo")  
  
str\_view(szoveg, pattern = "ar")



Az **anchor**-okkal azt lehet megadni, hogy a karakter string elején vagy végén szeretnénk egyezést találni. A string eleji anchor a ^, a string végi pedig a $.

str\_view("Dr. Doktor Dr.", pattern = "^Dr.")



str\_view("Dr. Doktor Dr.", pattern = "Dr.$")



Egy másik jellemző probléma, hogy olyan speciális karaktert akarunk leírni a regex kifejezésünkkel, ami amúgy a regex szintaxisban használt. Ilyen eset például a ., ami mint írásjel sokszor csak zaj, ám a regex kontextusban a “bármilyen karakter” megfelelője.

str\_view("Dr. Doktor Dr.", pattern = ".k.")



Ahhoz hogy magát az írásjelet jelöljük, a \\ -t kell elé rakni.

str\_view("Dr. Doktor Dr.", pattern = "\\.")



Néhány hasznos regex kifejezés:

* [:digit:] - számok (123)
* [:alpha:] - betűk (abc ABC)
* [:lower:] - kisbetűk (abc)
* [:upper:] - nagybetűk (ABC)
* [:alnum:] - betűk és számok (123 abc ABC)
* [:punct:] - központozás (.!?\(){})
* [:graph:] - betűk, számok és központozás (123 abc ABC .!?\(){})
* [:space:] - szóköz ( )
* [:blank:] - szóköz és tabulálás
* [:cntrl:] - kontrol karakterek (\n, \r, stb.)
* \* - bármi

# 4 Korpuszépítés és szövegelőkészítés

## 4.1 Szövegbeszerzés

A szövegbányászati elemzések egyik első lépése az elemzés alapjául szolgáló korpusz megépítése. A korpuszt alkotó szövegek beszerzésének egyik módja a webscarping, melynek során weboldalakról történik az információ kinyerése.

A scrapelést végezhetjük R-ben az rvest csomag segítségével. Fejezetünkben a scrapelésnek csupán néhány alaplépését mutatjuk meg, a folyamatról bővebb információ található például az alábbi oldalakon: <https://cran.r-project.org/web/packages/rvest/rvest.pdf>, <https://rvest.tidyverse.org>.

library(rvest)  
library(readr)  
library(dplyr)  
library(lubridate)  
library(stringr)  
library(quanteda)  
library(quanteda.textmodels)

A szükséges csomagok beolvasása után a read\_html() függvény segítségével az adott weboldal adatait kérjük le a szerverről. A read\_html() függvény argumentuma az adott weblap URL-je.

Ha például a poltextLAB projekt honlapjáról szeretnénk adatokat gyűjteni, azt az alábbi módon tehetjük meg:

r <- read\_html("https://poltextlab.tk.hu/hu")  
  
r  
#> {html\_document}  
#> <html lang="hu" class="no-js">  
#> [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...  
#> [2] <body class="index">\n\n\t<script>\n\t (function(i,s,o,g,r,a,m){i['Googl ...

Ezután a html\_nodes() függvény argumentumaként meg kell adnunk azt a HTML címkét vagy CSS azonosítót, ami a legyűjteni kívánt elemeket azonosítja a weboldalon. Ezeket az azonosítókat az adott weboldal forráskódjának megtekintésével tudhatjuk meg, amire a különböző böngészők különböző lehetőségeket kínálnak. Majd a html\_text() függvény segítségével megkapjuk azokat a szövegeket, amely az adott weblapon az adott azonosítóval rendelkeznek.

Példánkban a <https://poltextlab.tk.hu/hu> weboldalról azokat az információkat szeretnénk kigyűjteni, amelyek az <title> címke alatt szerepenek:

title <- read\_html("https://poltextlab.tk.hu/hu") %>% html\_nodes("title") %>% html\_text()  
  
title  
#> [1] "MTA TK Political and Legal Text Mining and Artificial Intelligence Laboratory (poltextLAB)"

A kigyűjtött információkat pedig ezután kiíratjuk egy csv fájlba.

write\_csv(title, "title.csv")

A web scraping során az egyik nehézség, ha a weboldal letiltja az automatikus letöltést, ezt kivédhetjük például különböző böngészőbővítmények segítségével, illetve a fejléc (header) vagy a user agent megváltoztatásával, de segíthet véletlenszerű proxy vagy VPN szolgáltatás használata is, valamint ha az egyes kérések között időt hagynunk. A weboldalakon legtöbbször a legyűjtött szövegekhez tartozó különböző metaadatok is szerepelnek (például egy parlamenti beszéd dátuma, az azt elmondó képviselő neve), melyeket érdemes a scarpelés során szintén összegyűjteni. A scrapelés során fontos figyelnünk arra, hogy később jól használható formában mentsük el az adatokat, például .csv,.json vagy .txt kiterjesztésekkel. A karakterkódolási problémák elkerülése érdekében érdemes UTF-8 vagy UTF-16-os kódolást alkalmazni, mivel ezek tartalmazzák a magyar nyelv ékezetes karaktereit is. A karakterkódolással kapcsolatosan hasznos további információk találhatóak az alábbi oldalon: <http://www.cs.bme.hu/~egmont/utf8>

Arra is van lehetőség, hogy az elemezni kívánt korpuszt papíron keletkezett, majd szkennelt és szükség szerint optikai karakterfelismerés (OCR, *Optical Character Recognition*) segítségével feldolgozott szövegekből építsük fel. Azonban mivel ezeket a feladatokat nem R-ben végezzük, ezekről itt nem szólunk bővebben. Az így beszerzett és .txt, vagy .csv fájllá alakított szövegekből való korpuszépítés a következő lépésekben megegyezik a weboldalakról gyűjtött szövegekével.

## 4.2 Szövegelőkészítés

Az elemzéshez vezető következő lépés a szövegelőkészítés, amit a szöveg tisztításával kell kezdenünk. A szövegtisztításnál mindig járjunk el körültekintően és az egyes lépéseket a kutatási kérdésünknek megfelelően tervezzük meg, a folyamat során pedig időnként végezzünk ellenőrzést, ezzel elkerülhetjük a kutatásunkhoz szükséges információk elvesztését.

Miután az elemezni kívánt szövegeinket beszereztük, majd a “Az adatok importálása és exportálása” alfejezetben leírtak szerint importáltuk, következhetnek az alapvető előfeldolgozási lépések, ezek közé tartozik például a scrapelés során a kopuszba került html címkék, számok és egyéb zajok (például a speciális karakterek, írásjelek) eltávolítása a korpuszból, valamint a kisbetűsítés, a tokenizálás, a szótövezés és a stopszavazás.

### 4.2.1 String műveletek

A stringr csomag segítségével először eltávolíthatjuk a felesleges html címkéket a korpuszból. Ehhez először létrehozzuk a text1 nevű objektumot ami egy karaktervektorból áll.

text1 <- c("MTA TK", "<font size='6'> Political and Legal Text Mining and Artificial Intelligence Laboratory (poltextLAB)")  
  
text1  
#> [1] "MTA TK"   
#> [2] "<font size='6'> Political and Legal Text Mining and Artificial Intelligence Laboratory (poltextLAB)"

Majd a str\_replace\_all()függvény segítségével eltávolítjuk két html címke közötti szövegrészt. Ehhez a függvény argumentumában létrehozunk egy regex kifejezést, aminek segítségével a függvény minden <> közötti szövegrészt üres karakterekre cserél. Ezután a str\_to\_lower()mindent kisbetűvé konvertál, majd a str\_trim()eltávolítja a szóközöket a karakterláncok elejéről és végéről.

text1 %>%  
 str\_replace\_all(pattern = "<.\*?>", replacement = "") %>%  
 str\_to\_lower() %>%  
 str\_trim()  
#> [1] "mta tk"   
#> [2] "political and legal text mining and artificial intelligence laboratory (poltextlab)"

### 4.2.2 Tokenizálás, szótövezés, kisbetűsítés és a stopszavak eltávolítása

Az előkészítés következő lépésében tokenizáljuk, azaz egységeire bontjuk az elemezni kívánt szöveget, a tokenek így pedig az egyes szavakat vagy kifejezéseket fogják jelölni. Ennek eredményeként kapjuk meg az n-gramokat, amik a vizsgált egységek (számok, betűk, szavak, kifejezések) n-elemű sorozatát alkotják.

A következőkben a “Példa az előkészítésre” mondatot bontjuk először tokenekre a tokens() függvénnyel, majd a tokeneket a tokens\_tolower() segítségével kisbetűsítjük, a tokens\_wordstem() függvénnyel pedig szótövezzük. Végezetül a quanteda csomagban található magyar nyelvű stopszótár segítségével, elvégezzük a stopszavak eltávolítását.Ehhez először létrehozzuk az sw elnevezésű karaktervektort a magyar stopszvakból. A head() függvény segítségével belenézhetünk a szótárba, és a console-ra kiírathatjuk a szótár első hat szavát. Végül a tokens\_remove()segítségével eltávolítjuk a stopszavakat.

text <- "Példa az elokészítésre"  
  
toks <- tokens(text)  
  
toks <- tokens\_tolower(toks)  
  
toks <- tokens\_wordstem(toks)  
  
toks  
#> Tokens consisting of 1 document.  
#> text1 :  
#> [1] "példa" "az" "elokészítésr"  
  
sw <- stopwords("hungarian")  
  
head(sw)  
#> [1] "a" "ahogy" "ahol" "aki" "akik" "akkor"  
  
tokens\_remove(toks, sw)  
#> Tokens consisting of 1 document.  
#> text1 :  
#> [1] "példa" "elokészítésr"

#### 4.2.2.1 Stemmelés vagy lemmatizálás

Ezt követi a szótövezés lépése, melynek során az alkalmazott stemmelő algoritmus egyszerűen levágja a szavak összes toldalékát, a képzőket, jelzőket és ragokat. A stemmelés helyett alkalmazhatunk lemmatizálást, melynek során a szavakat a szótári alakjukra formáljuk. A stemming és lemmatizálás közötti különbség abban rejlik, hogy a szótövezés során csupán eltávolítjuk a szavak toldalékként azonosított végződéseit, hogy ugyanannak a szónak különböző megjelenési formáit közös törzsre redukáljuk, míg a lemmatizálás esetében rögtön az értelmes, szótári formát kapjuk vissza. A két módszer közötti választás a kutatási kérdés alapján meghozott kutatói döntésen alapul ([Grimmer and Stewart 2013](#ref-grimmer2013text)).

##### 4.2.2.1.1 Lemmatizálás

Az alábbi példában egyetlen szó különböző alakjainak szótári alakra hozásával szemléltetjük a lemmatizáslás működését.

Ehhez először a text1 nevű objektumban tároljuk a lemmatizálni kívánt szöveget, majd tokenizáljuk és eltávolítjuk a központozást. Ezután definiáljuk azt a megfelelő szótövet és azt, hogy mely szavak alakjait szeretnénk erre a szótőre egységesíteni majd a rep() függvény segítségével elvégezzük a lemmatizálást, amely a korábban definiált szólakokat az általunk megadott szótári alakkal helyettesíti. Hosszabb szövegek lemmatizálásához előre létrehozott szótárakat használhatunk, ilyen például a WordNet, ami magyar nyelven is elérhető: <https://github.com/mmihaltz/huwn> A magyar nyelvű szövegek lemmatizálását elvégezhetjük a szövegek R-be való beolvasása előtt is a magyarlancnyelvi elemző segítségével, melyről a “Természetes-nyelv feldolgozás (NLP) és névelemfelismerés” című fejezetben szólunk részletesebben.

text1 <- "Példa az elokészítésre. Az elokészítést a szövetisztítással kell megkezdenünk. Az elokészített korpuszon elemzést végzünk"  
  
toks1 <- tokens(text1, remove\_punct = TRUE)  
  
elokeszites <- c("elokészítésre", "elokészítést", "elokészített")  
  
lemma <- rep("elokészítés", length(elokeszites))  
  
toks1 <- tokens\_replace(toks1, elokeszites, lemma, valuetype = "fixed")  
  
toks1  
#> Tokens consisting of 1 document.  
#> text1 :  
#> [1] "Példa" "az" "elokészítés"   
#> [4] "Az" "elokészítés" "a"   
#> [7] "szövetisztítással" "kell" "megkezdenünk"   
#> [10] "Az" "elokészítés" "korpuszon"   
#> [ ... and 2 more ]

##### 4.2.2.1.2 Stemmelés

A fenti text1 objektumban tárolt szöveg stemmelését az alábbiak szerint tudjuk elvégezni. Megvizsgálva az előkészítés különböző alakjainak lemmatizált és stemmelt változatát jól láthatjuk a két módszer közötti különbséget.

text1 <- "Példa az elokészítésre. Az elokészítést a szövetisztítással kell megkezdenünk. Az elokészített korpuszon elemzést végzünk"  
  
toks2 <- tokens(text1, remove\_punct = TRUE)  
  
toks2 <- tokens\_wordstem(toks2)  
  
toks2  
#> Tokens consisting of 1 document.  
#> text1 :  
#> [1] "Példa" "az" "elokészítésr" "Az"   
#> [5] "elokészítést" "a" "szövetisztításs" "kell"   
#> [9] "megkezdenünk" "Az" "elokészített" "korpuszon"   
#> [ ... and 2 more ]

### 4.2.3 Dokumentum kifejezés mátrix (DTM)

A szövegbányászati elemzések nagy részéhez szükségünk van arra, hogy a szövegeinkből dokumentum kifejezés matrix-ot (DTM), vagy dokumentum feature matrxi-ot (DFM) hozzunk létre. Ezzel a lépéssel alakítjuk a szövegeinket számokká, ami lehetővé teszi, hogy utána különböző statisztikai műveleteket végezzünk velük.

A dokumentum kifejezés mátrix minden sora egy dokumentum, minden oszlopa egy kifejezés, az oszlopokban szereplő változók pedig az egyes kifejezések számát mutatják meg az egyes dokumentumokban. A legtöbb DTM ritka mátrix, mivel a legtöbb dokumentum és kifejezés párosítása nem történik meg, mivel a kifejezések nagy része csak néhány dokumentumban szerepel, ezek értéke nulla lesz.

Az alábbi példában három egy-egy mondatos dokumentumon szemléltetjük a fentieket. A korábban megismert módon előkészítjük, azaz kisbetűsítjük, stemmeljük és stopszavazzuk a dokumentumokat, majd létrehozzuk belőlük a dokumentum kifejezés mátrixot.

text <- c(  
 d1 = "Ez egy példa az előfeldolgozásra",  
 d2 = "Egy másik lehetséges példa",  
 d3 = "Ez pedig egy harmadik példa"  
)  
  
dtm <- dfm(  
 text,  
 tolower = TRUE, stem = TRUE,  
 remove = stopwords("hungarian")  
)  
  
dtm  
#> Document-feature matrix of: 3 documents, 4 features (50.0% sparse).  
#> features  
#> docs példa elofeldolgozásra lehetség harmadik  
#> d1 1 1 0 0  
#> d2 1 0 1 0  
#> d3 1 0 0 1

Egy másik szövegbányászati megközelítés a mátrixot nem DTM-nek, hanem DFM-nek nevezi, például a quanteda csomag használata során nem DTM-et, hanem DFM-et kell létrehoznunk.

text <- c(  
 d1 = "Ez egy példa az elofeldolgozásra",  
 d2 = "Egy másik lehetséges példa",  
 d3 = "Ez pedig egy harmadik példa"  
)  
  
dfm <- dfm(  
 text,  
 tolower = TRUE, stem = TRUE,  
 remove = stopwords("hungarian")  
)  
  
dfm  
#> Document-feature matrix of: 3 documents, 4 features (50.0% sparse).  
#> features  
#> docs példa elofeldolgozásra lehetség harmadik  
#> d1 1 1 0 0  
#> d2 1 0 1 0  
#> d3 1 0 0 1

### 4.2.4 Súlyozás

A dokumentum kifejezés mátrix lehet egy egyszerű bináris mátrix, ami csak azt az információt tartalmazza, hogy egy adott szó előfordul-e egy adott dokumentumban. Míg az egyszerű bináris mátrixban ugyanakkora súlya van egy szónak ha egyszer és ha tízszer szerepel, készíthetünk olyan mátrixot is, ahol egy szónak annál nagyobb a súlya egy dokumentumban, minél többször fordul elő. A szógyakoriság (term frequency, TM) szerint súlyozott TD mátrixnál azt is figyelembe vesszük, hogy az adott szó hány dokumentumban szerepel. Minél több dokumentumban szerepel egy szó, annál kisebb a jelentősége. Ilyen szavak például a névelők, amelyek sok dokumentumban előfordulnak ugyan, de nem sok tartalmi jelentőséggel bírnak. Két szó közül általában az a fontosabb, amelyik koncentráltan, kevés dokumentumban, de azokon belül nagy gyakorisággal fordul elő. A dokumentum gyakorisági érték (document frequency, df) egy szó ritkaságát jellemzi egy korpuszon belül, azaz megadja, hogy mekkora megkülönböztető ereje van egy szónak a dokumentum tartalmára vonatkozóan. A súlyozási sémákban általában a dokumentum gyakorisági érték inverzével számolnak (inverse document frequency, idf) ez a leggyakrabban használt td-idf súlyozás (term frequency & inverse document frequency. Az így súlyozott TD mátrix egy-egy cellájában található érték azt mutatja, hogy egy adott szónak mekkora a jelentősége egy adott dokumentumban. A tf -idf súlyozás értéke tehát magas azon szavak esetén, amelyek az adott dokumentumban gyakran fordulnak elő, míg a teljes korpuszban ritkán, alacsonyabb azon szavak esetén, amelyek az adott dokumentumban ritkábban, vagy a korpuszban gyakrabban fordulnak elő és kicsi azon szavaknál, amelyek a korpusz lényegében összes dokumentumában előfordulnak ([Tikk 2007, 33–37](#ref-tikkSzovegbanyaszat2007) o.)

Az alábbiakban az 1999-es törvényszövegeken szemléltetjük hogy egy 125 dokumentumból létrehozott mátrix segítségével milyen alapvető statisztikai műveleteket végezhetünk.[[9]](#footnote-74)

Ehhez először importáljuk a törvények egy .csv kiterjesztésű fájlból. A read\_csv() használatának az az előnye, hogy alapbeállításként UTF-8 formátumban importálja be a szöveges oszlopokat.

lawtext\_df <- read\_csv("data/lawtext\_1999.csv")

Majd az importált fájlokból létrehozzuk a korpusz lawtext\_corpus néven. Ezt követi a dokumentum kifejezés mátrix kialakítása (mivel a quanteda csomaggal dolgozunk, dfm mátrixot hozunk létre), és ezzel egy lépésben, elvégezzük az alapvető szövegtisztitó lépéseket is.

lawtext\_corpus <- corpus(lawtext\_df)  
  
lawtext\_dfm <- dfm(  
 lawtext\_corpus,  
 tolower = TRUE,  
 remove = stopwords("hungarian"),  
 stem = TRUE,  
 remove\_punct = TRUE,  
 remove\_symbols = TRUE,  
 remove\_numbers = TRUE  
)

A topfeatures függvény segítségével megnézhetjük a mátrix leggyakoribb szavait a függvény argumentumában a dokumentum kifejezés mátrix nevét és a kívánt kifejezésszámot megadva.

topfeatures(lawtext\_dfm, 15)  
#> the of szerzodo to b ha and kiadások   
#> 7902 5665 3619 3290 2831 2794 2712 2447   
#> törvéni in következo muködési or évi is   
#> 2385 2253 2178 2038 2034 1908 1864

Mivel látható, hogy a szövegekben sok angol kifejezés is volt egy következő lépcsőben, az angol stopszavakat is eltávolítjuk.

lawtext\_dfm\_2 <- dfm(lawtext\_dfm, remove = stopwords("english"))

Majd ismét megnézzük a leggyakoribb 15 kifejezést.

topfeatures(lawtext\_dfm\_2, 15)  
#> szerzodo b ha kiadások törvéni   
#> 3619 2831 2794 2447 2385   
#> következo muködési évi állam c   
#> 2178 2038 1908 1718 1713   
#> meghatározott költségveté államban lép fél   
#> 1654 1637 1622 1616 1533

Ezután tf-idf súlyozású statisztikát készítünk, a dokumentum kifejezés mátrix alapján. Ehhez először létrehozzuk a lawtext\_tfidf nevű objektumot, majd a textstat\_frequency függvény segítségével, és kilistázzuk annak első 10 elemét.

lawtext\_tfidf <- dfm\_tfidf(lawtext\_dfm\_2)  
  
textstat\_frequency(lawtext\_tfidf, force = TRUE, n = 10)  
#> feature frequency rank docfreq group  
#> 1 kiadások 2120.2303 1 17 all  
#> 2 felhalmozási 1448.3465 2 7 all  
#> 3 szerzodo 1378.5012 3 52 all  
#> 4 költségveté 1302.8556 4 20 all  
#> 5 shall 1291.1619 5 14 all  
#> 6 államban 1223.7785 6 22 all  
#> 7 részes 1155.9688 7 13 all  
#> 8 muködési 1101.7581 8 36 all  
#> 9 articl 967.8961 9 14 all  
#> 10 parti 845.2246 10 20 all

# 5 Információ-visszakeresés és információkinyerés, szövegek reprezentálása a vektortérben, leíró statisztika

A szövegbányászati feladatok két altípusa a keresés és a rendszerezés. A keresés során olyan szövegeket keresünk, amelyekben egy adott kifejezés előfordul, a rendszerezés során pedig a szövegeket hasonlítjuk össze egymással és egy előre megadott, vagy egy előzetesen nem ismert kategóriarendszer csoportjaihoz soroljuk őket. Az információ-visszakeresés (information retrieval), ami például a webes keresőprogramok egyik jellemző tevékenysége, során a cél, hogy a korpuszból visszakeressük a kereső információigénye szempontjából releváns információkat, mely keresés alapulhat metaadatokon vagy teljes szöveges indexelésen ([Tikk 2007](#ref-tikkSzovegbanyaszat2007); [Russel and Norvig 2005, 742.o](#ref-russelMestersegesIntelligencia2005)). Az információkinyerés (information extraction) esetén a cél, hogy a strukturálatlan szövegekből strukturált adatokat állítsunk elő. Azaz az információkinyerés során nem a felhasználó által keresett információt keressük meg és lokalizáljuk, hanem az adott kérdés szempontjából releváns információkat gyűjtjük ki a dokumentumokból. aAz információkinyerés alternatív megoldása segítségével már képesek lehetünk a kifejezések közötti kapcsolatok elemzésére, tendenciák és minták felismerésére és az információk összekapcsolás révén új információk létrehozására. Azaz a segítségével strukturálatlan szövegekből is előállíthatunk strukturált információkat ([Kwartler 2017](#ref-kwartlerTextMiningPractice2017); [Schütze, Manning, and Raghavan 2008](#Xa269f770aa4a47943f7101d538748afa012fcd7); [Tikk 2007, 63–81.o](#ref-tikkSzovegbanyaszat2007)).

## 5.1 A szövegek reprezentálása a vektortérben - szózsák modell

A szövegbányászati vizsgálatok során folyó szövegek, azaz strukturálatlan vagy részben strukturált dokumentumok elemzésére kerül sor, melyekből a kutatási kérdéseink szempontjából releváns, látens összefüggéseket nyerünk ki, amelyek már strukturált szerkezetűek. A dokumentumok reprezentálásnak három legelterjedtebb módja a halmazelmélet alapú, az algebrai és a valószínűségi modell. A halmazelméleti modellek a dokumentumok hasonlóságát halmazelmélet, a valószínűségi modellek pedig feltételes valószínűségi becslés alapján határozzák meg. Az algebrai modellek a dokumentumokat vektorként vagy mátrixként ábrázolják és algebrai műveletek segítségével hasonlítják össze. A vektortérmodell sokdimenziós vektortérben ábrázolja a dokumentumokat, úgy hogy a dokumentumokat vektorokkal reprezentálja, a vektortér dimenziói pedig a dokumentumok összességében előforduló egyedi szavak. A modell alkalmazása során azok a dokumentumok hasonlítanak egymásra, amelyeknek a szókészlete átfedi egymást, és a hasonlóság mértéke az átfedéssel arányos. A vektortérmodellben a dokumentumgyűjteményt a szó-dokumentum mátrixszal (term-document matrix) reprezentáljuk, a mátrixban a sorok száma megegyezik az egyedi szavak számával, az oszlopokat pedig a dokumentumvektorok alkotják. Az egyedi szavak összességét szórátnak nevezzük. Mivel mátrixban az egyedi szavak száma általában igen nagy, ezért a mátrix hatékony kezeléséhez annak mérete különböző eljárásokkal csökkenthető. Fontos tudni, hogy a dokumentumok vektortér reprezentációjában a szavak szövegen belüli sorrendjére és pozíciójára vonatkozó információ nem található meg ([Russel and Norvig 2005, 742–44](#ref-russelMestersegesIntelligencia2005) o.; [Kwartler 2017](#ref-kwartlerTextMiningPractice2017); [Welbers, Van Atteveldt, and Benoit 2017](#ref-welbersTextAnalysis2017)). A vektortérmodellt szózsák (bag of words) modellnek is nevezzük, melynek segítségével a fent leírtak szerint az egyes szavak gyakoriságát vizsgálhatjuk meg egy adott korpuszon belül.

A szó-dokumentum mátrix lehet egy egyszerű bináris mátrix, ami csak azt az információt tartalmazza, hogy egy adott szó előfordul-e egy adott dokumentumban. Míg az egyszerű bináris mátrixban ugyanakkora súlya van egy szónak ha egyszer és ha tízszer szerepel, készíthetünk olyan mátrixot is, ahol egy szónak annál nagyobb a súlya egy dokumentumban, minél többször fordul elő. Emellett a mátrixot a dokumentumok hossza szerint is normálhatjuk. Az így súlyozott mátrixban nem azt vizsgáljuk, hogy egy szó hányszor egy dokumentumban, hanem ezt a számot viszonyítjuk az adott dokumentum szavainak a számához. A szógyakoriság (term frequency, TM) szerint súlyozott TD mátrixnál azt is figyelembe vesszük, hogy az adott szó hány dokumentumban szerepel. Minél több dokumentumban szerepel egy szó, annál kisebb a jelentősége. Ilyen szavak például a névelők, amelyek sok dokumentumban előfordulnak ugyan, de nem sok tartalmi jelentőséggel bírnak. Két szó közül általában az a fontosabb, amelyik koncentráltan, kevés dokumentumban, de azokon belül nagy gyakorisággal fordul elő. A dokumentum gyakorisági érték (document frequency, df) egy szó ritkaságát jellemzi egy korpuszon belül, azaz megadja, hogy mekkora megkülönböztető ereje van egy szónak a dokumentum tartalmára vonatkozóan. A súlyozási sémákban általában a dokumentum gyakorisági érték inverzével számolnak (inverse document frequency, idf) ez a leggyakrabban használt td-idf súlyozás (term frequency & inverse document frequency. Az így súlyozott TD mátrix egy-egy cellájában található érték azt mutatja, hogy egy adott szónak mekkora a jelentősége egy adott dokumentumban. A tf -idf súlyozás értéke tehát magas azon szavak esetén, amelyek az adott dokumentumban gyakran fordulnak elő, míg a teljes korpuszban ritkán, alacsonyabb azon szavak esetén, amelyek az adott dokumentumban ritkábban, vagy a korpuszban gyakrabban fordulnak elő és kicsi azon szavaknál, amelyek a korpusz lényegében összes dokumentumában előfordulnak ([Tikk 2007, 33–37](#ref-tikkSzovegbanyaszat2007) o.; [Grimmer and Stewart 2013](#ref-grimmer2013text); [Welbers, Van Atteveldt, and Benoit 2017](#ref-welbersTextAnalysis2017))

## 5.2 Leíró statisztika

Fejezetünkben nyolc véletlenszerűen kiválasztott magyar miniszterelnöki beszéd vizsgálatát végezzük el, amihez az alábbi csomagokat használjuk:

library(readtext)  
library(dplyr)  
library(lubridate)  
library(stringr)  
library(ggplot2)  
library(quanteda)  
library(GGally)  
library(tidytext)  
library(tidyverse)

Első lépésben a már ismertetett módon a readtext () segítségével beolvassuk a beszédek .txt formátumú változatát, utf-8 karakterkódolással.

texts <- readtext("data/mineln/\*.txt", encoding = "UTF-8")  
  
head(texts)  
#> readtext object consisting of 6 documents and 0 docvars.  
#> # Description: df[,2] [6 x 2]  
#> doc\_id text   
#> \* <chr> <chr>   
#> 1 antall\_jozsef\_1990.txt "\"Elnök Úr! \"..."  
#> 2 bajnai\_gordon\_2009.txt "\"Tisztelt E\"..."  
#> 3 boross\_peter\_1993.txt "\"Elnök Úr! \"..."  
#> 4 gyurcsany\_ferenc\_2005.txt "\"Mélyen tis\"..."  
#> 5 horn\_gyula\_1994.txt "\"Köszönöm, \"..."  
#> 6 medgyessy\_peter\_2002.txt "\"Mélyen tis\"..."

Ezt követően az “Adatkezelés R-ben” fejezetben ismertetett mutate() függvény használatával két csoportra osztjuk a beszédeket. Ehhez először a string\_extract()segítségével meghatározzuk, hogy a kettéosztáshoz használni kívánt új változó a doc\_id legyen a [^\\.]\* regex segítségével leválasztva arról a .txt kiterjesztést, majd a str\_sub() függvénnyel megmondjuk, hogy a miniszterelnökök neve a doc\_id hátulról számított hatodik karakteréig tart. Ezután kialakítjuk a két csoportot, azaz az if\_else() segítségével meghatározzuk, hogy ha “antall\_jozsef,” “boross\_peter,” “orban\_viktor” beszédeiről van szó azokat a jobb csoportba tegye, a maradékot pedig a bal csoportba.

Majd azt is meghatározzuk, hogy melyik beszédnek mi a dátuma. Ehhez szintén a str\_sub() függvényt használjuk, majd a lubridate segítségével alakítjuk ki a kívánt dátumformátumot[[10]](#footnote-81).

Majd a glimpse() függvény segítségével megtekintjük, hogy milyen változtatásokat végeztünk az adattáblánkon. Láthatjuk, hogy míg korábban 8 dokumentumunk és 2 változónk volt, az átalakítás eredményeként a 8 dokumentum mellett már 5 változót találunk. Ezzel a lépéssel tehát kialakítottuk azokat a változókat, amelyekre az elemzés során szükségünk lesz.

texts <- texts %>% mutate(doc\_id = str\_extract(doc\_id, "[^\\.]\*"), mineln = str\_sub(doc\_id,   
 end = -6), wing = if\_else(mineln %in% c("antall\_jozsef", "boross\_peter", "orban\_viktor"),   
 "jobb", "bal"))  
  
texts$year <- str\_sub(texts$doc\_id, start = -2) %>% str\_c("-01-01") %>% lubridate::ymd() %>%   
 lubridate::year()  
  
glimpse(texts)  
#> Rows: 8  
#> Columns: 5  
#> $ doc\_id <chr> "antall\_jozsef\_1990", "bajnai\_gordon\_2009", "boross\_peter\_19...  
#> $ text <chr> "Elnök Úr! Tisztelt Országgyulés! Hölgyeim és Uraim! Honfitá...  
#> $ mineln <chr> "antall\_jozsef", "bajnai\_gordon", "boross\_peter", "gyurcsany...  
#> $ wing <chr> "jobb", "bal", "jobb", "bal", "bal", "bal", "jobb", "jobb"  
#> $ year <dbl> 1990, 2009, 1993, 2005, 1994, 2002, 1995, 2018

Ezt követően a további lépések elvégzéséhez létrehozzuk a quanteda korpuszt, majd a summary() függvény segítségével megtekinthetjük a korpusz alapvető statisztikai jellemzőit, láthatjuk például, hogy az egyes dokumentumok hány tokenből vagy mondatból állnak.

corpus\_mineln <- corpus(texts)  
  
summary(corpus\_mineln)  
#> Corpus consisting of 8 documents, showing 8 documents:  
#>   
#> Text Types Tokens Sentences mineln wing year  
#> antall\_jozsef\_1990 3745 9408 431 antall\_jozsef jobb 1990  
#> bajnai\_gordon\_2009 1391 3277 201 bajnai\_gordon bal 2009  
#> boross\_peter\_1993 1552 3170 179 boross\_peter jobb 1993  
#> gyurcsany\_ferenc\_2005 2963 10267 454 gyurcsany\_ferenc bal 2005  
#> horn\_gyula\_1994 1704 4372 226 horn\_gyula bal 1994  
#> medgyessy\_peter\_2002 1021 2362 82 medgyessy\_peter bal 2002  
#> orban\_viktor\_1995 1810 4287 212 orban\_viktor jobb 1995  
#> orban\_viktor\_2018 933 1976 126 orban\_viktor jobb 2018

Mivel az elemzés során a korpuszon belül két csoportra osztva szeretnénk összehasonlításokat tenni, az alábbiakban két alkorpuszt alakítunk ki.

mineln\_jobb <- corpus\_subset(corpus\_mineln, mineln %in% c("antall\_jozsef", "boross\_peter",   
 "orban\_viktor"))  
  
mineln\_bal <- corpus\_subset(corpus\_mineln, mineln %in% c("horn\_gyula", "gyurcsany\_ferenc",   
 "medgyessy\_peter", "bajnai\_gordon"))  
  
summary(mineln\_jobb)  
#> Corpus consisting of 4 documents, showing 4 documents:  
#>   
#> Text Types Tokens Sentences mineln wing year  
#> antall\_jozsef\_1990 3745 9408 431 antall\_jozsef jobb 1990  
#> boross\_peter\_1993 1552 3170 179 boross\_peter jobb 1993  
#> orban\_viktor\_1995 1810 4287 212 orban\_viktor jobb 1995  
#> orban\_viktor\_2018 933 1976 126 orban\_viktor jobb 2018  
  
summary(mineln\_bal)  
#> Corpus consisting of 4 documents, showing 4 documents:  
#>   
#> Text Types Tokens Sentences mineln wing year  
#> bajnai\_gordon\_2009 1391 3277 201 bajnai\_gordon bal 2009  
#> gyurcsany\_ferenc\_2005 2963 10267 454 gyurcsany\_ferenc bal 2005  
#> horn\_gyula\_1994 1704 4372 226 horn\_gyula bal 1994  
#> medgyessy\_peter\_2002 1021 2362 82 medgyessy\_peter bal 2002

A korábban létrehozott “jobb” és “bal” változó segítségével nem csak az egyes dokumentumokat, hanem a két csoportba sorolt beszédeket is összehasonlíthatjuk egymással.

summary(corpus\_mineln) %>%  
 group\_by(wing) %>%  
 summarise(mean\_wordcount = mean(Tokens), std\_dev = sd(Tokens), min\_wordc = min(Tokens), max\_wordc = max(Tokens))  
#> # A tibble: 2 x 5  
#> wing mean\_wordcount std\_dev min\_wordc max\_wordc  
#> <chr> <dbl> <dbl> <int> <int>  
#> 1 bal 5070. 3561. 2362 10267  
#> 2 jobb 4710. 3271. 1976 9408

A textstat\_collocations() függvény segítségével együtt előforduló szókapcsolatokat kereshetünk. A függvény argumentumai közül a size a szókapcsolatok hossza, a min\_count pedig a minimális előfordulásuk száma. Miután a szókapcsolatokat megkeressük a korábban már megismert head() függvény segítségével megnézhetünk közülük tetszőleges számút[[11]](#footnote-83).

corpus\_mineln %>%  
 textstat\_collocations(  
 size = 3,  
 min\_count = 6  
 ) %>%  
 head(n = 10)  
#> collocation count count\_nested length lambda z  
#> 1 a kormány a 30 0 3 1.7266498 3.5500939  
#> 2 az új kormány 13 0 3 4.7126130 2.9870558  
#> 3 az a politika 6 0 3 3.9239765 2.4659912  
#> 4 a kormány az 6 0 3 2.6826277 1.7954739  
#> 5 a száz lépés 9 0 3 3.5817972 1.5956086  
#> 6 a magyar gazdaság 14 0 3 2.5135668 1.5757358  
#> 7 ez a program 9 0 3 1.9371894 1.2837433  
#> 8 tisztelt hölgyeim és 31 0 3 2.2327116 0.9596371  
#> 9 hogy ez a 10 0 3 0.4617372 0.9032165  
#> 10 hogy a magyar 18 0 3 0.6903168 0.7832561

A szókapcsolatok listázásánál is láthattuk, hogy a korpuszunk még minden szót tartalmaz, ezért találtunk például “hogy ez a” összetételt. A következőkben eltávolítjuk az ilyen funkció nélküli stopszavakat a korpuszból, amihez saját stopszólistát használunk. Először beolvassuk és egy custom\_stopwords nevű objektumban tároljuk a stopszavakat, majd a tokens() függvény segítségével tokenizáljuk a korpuszt és a tokens\_select() használatával eltávolítjuk a stopszavakat.

Ha ezután újra megnézzük a kollokációkat, jól látható a stopszavak eltávolításának eredménye:

custom\_stopwords <- readLines("data/stopwords\_hun.txt", encoding = "UTF-8")  
  
corpus\_mineln %>%  
 tokens() %>%  
 tokens\_select(pattern = custom\_stopwords, selection = "remove") %>%  
 textstat\_collocations(  
 size = 3,  
 min\_count = 6  
 ) %>%  
 head(n = 10)  
#> collocation count count\_nested length lambda z  
#> 1 taps MSZP soraiból 7 0 3 -1.848559 -1.003837  
#> 2 tisztelt hölgyeim uraim 31 0 3 -3.217896 -1.087495  
#> 3 taps kormánypártok soraiban 13 0 3 -1.884367 -1.102199  
#> 4 közbeszólás fidesz soraiból 12 0 3 -4.371498 -1.949939  
#> 5 taps MSZP soraiban 9 0 3 -4.711439 -2.780059

A korpuszon további elemzése előtt fontos, hogy ne csak a stopszavakat távolítsuk el, hanem az egyéb alapvető szövegtisztító lépéseket is elvégezzük. Azaz a tokens\_select() segítségével eltávolítsuk a számokat, a központozást, az elválasztó karaktereket, mint például a szóközöket, tabulátorokat, sortöréseket. Ezután a tokens\_ngrams() segítségével ngramokat hozunk létre a tokenekből, majd kialakítjuk a dokumentum kifejezés mátrixot (dfm) és elvégezzük a tf-idf szerinti súlyozást. A dfm\_tfidf()függvény kiszámolja a dokumentum gyakoriság inverz súlyozását. A függvény alapértelmezés szerint a normalizált kifejezések gyakoriságát használja a dokumentumon belüli relatív kifejezés gyakoriság helyett, ezt írjuk felül a schem\_tf = "prop" használatával. Végül a textstat\_frequency() segítségével gyakorisági statisztikát készíthetünk a korábban meghatározott (példánkban két és három tagú) ngramokról.

corpus\_mineln %>%  
 tokens(  
 remove\_numbers = TRUE,   
 remove\_punct = TRUE,   
 remove\_separators = TRUE  
 ) %>%  
 tokens\_select(pattern = custom\_stopwords, selection = "remove") %>%  
 tokens\_ngrams(n = 2:3) %>%  
 dfm() %>%  
 dfm\_tfidf(scheme\_tf = "prop") %>%  
 textstat\_frequency(n = 10, force = TRUE)  
#> feature frequency rank docfreq group  
#> 1 tisztelt\_hölgyeim 0.002788904 1 4 all  
#> 2 tisztelt\_hölgyeim\_uraim 0.002788904 1 4 all  
#> 3 fordítsanak\_hátat 0.002439685 3 1 all  
#> 4 fidesz\_soraiból 0.002151922 4 1 all  
#> 5 taps\_mszp 0.001757918 5 2 all  
#> 6 magyarország\_európa 0.001457380 6 1 all  
#> 7 tisztelt\_képviselotársaim 0.001439932 7 2 all  
#> 8 kormánypártok\_soraiban 0.001430247 8 2 all  
#> 9 taps\_kormánypártok 0.001293617 9 2 all  
#> 10 taps\_kormánypártok\_soraiban 0.001293617 9 2 all

## 5.3 A szövegek lexikai diverzitása

Az alábbiakban a korpuszunkat alkotó szövegek lexikai diverzitását vizsgáljuk. Ehhez a quanteda csomag textstat\_lexdiv()függvényét használjuk. Mivel ez a függvény dfm-et elemez, először a corpus\_mineln nevű korpuszunkból létrehozzuk a mineln\_dfm nevű dfm-et, amelyen elvégezzük a korábban már megismert alapvető tisztító lépéseket. A textstat\_lexdiv() függvény eredménye szintén egy dfm, így azt arrange()parancs argumentumában a desc megadásával csökkenő sorba is rendezhetjük. Atextstat\_lexdiv() különböző indexek segítségével számítja ki a szövegek lexikai különbözőséget, példánkban a CTTR indexet használjuk[[12]](#footnote-86).

mineln\_dfm <- corpus\_mineln %>%  
 tokens(remove\_punct = TRUE, remove\_separators = TRUE, remove\_hyphens = TRUE) %>%  
 dfm(remove = custom\_stopwords)  
  
mineln\_dfm %>%  
 textstat\_lexdiv(measure = "CTTR") %>%  
 arrange(desc(CTTR))  
#> document CTTR  
#> 1 antall\_jozsef\_1990 32.99078  
#> 2 gyurcsany\_ferenc\_2005 26.14422  
#> 3 orban\_viktor\_1995 23.35548  
#> 4 horn\_gyula\_1994 22.25547  
#> 5 boross\_peter\_1993 21.98656  
#> 6 bajnai\_gordon\_2009 19.93214  
#> 7 medgyessy\_peter\_2002 16.81246  
#> 8 orban\_viktor\_2018 16.24532

A kiszámolt értéket hozzáadhatjuk a dfm-hez is.

dfm\_lexdiv <- mineln\_dfm  
  
cttr\_score <- unlist(textstat\_lexdiv(dfm\_lexdiv, measure = "CTTR")[, 2])  
  
docvars(dfm\_lexdiv, "cttr") <- cttr\_score  
  
docvars(dfm\_lexdiv)  
#> mineln wing year cttr  
#> 1 antall\_jozsef jobb 1990 32.99078  
#> 2 bajnai\_gordon bal 2009 19.93214  
#> 3 boross\_peter jobb 1993 21.98656  
#> 4 gyurcsany\_ferenc bal 2005 26.14422  
#> 5 horn\_gyula bal 1994 22.25547  
#> 6 medgyessy\_peter bal 2002 16.81246  
#> 7 orban\_viktor jobb 1995 23.35548  
#> 8 orban\_viktor jobb 2018 16.24532

A fenti elemzést elvégezhetjük úgy is, hogy valamennyi indexálást egyben megkapjuk. Ehhez a textstat\_lexdiv() függvény argumentumába a measure = "all" kifejezést kell megadnunk.

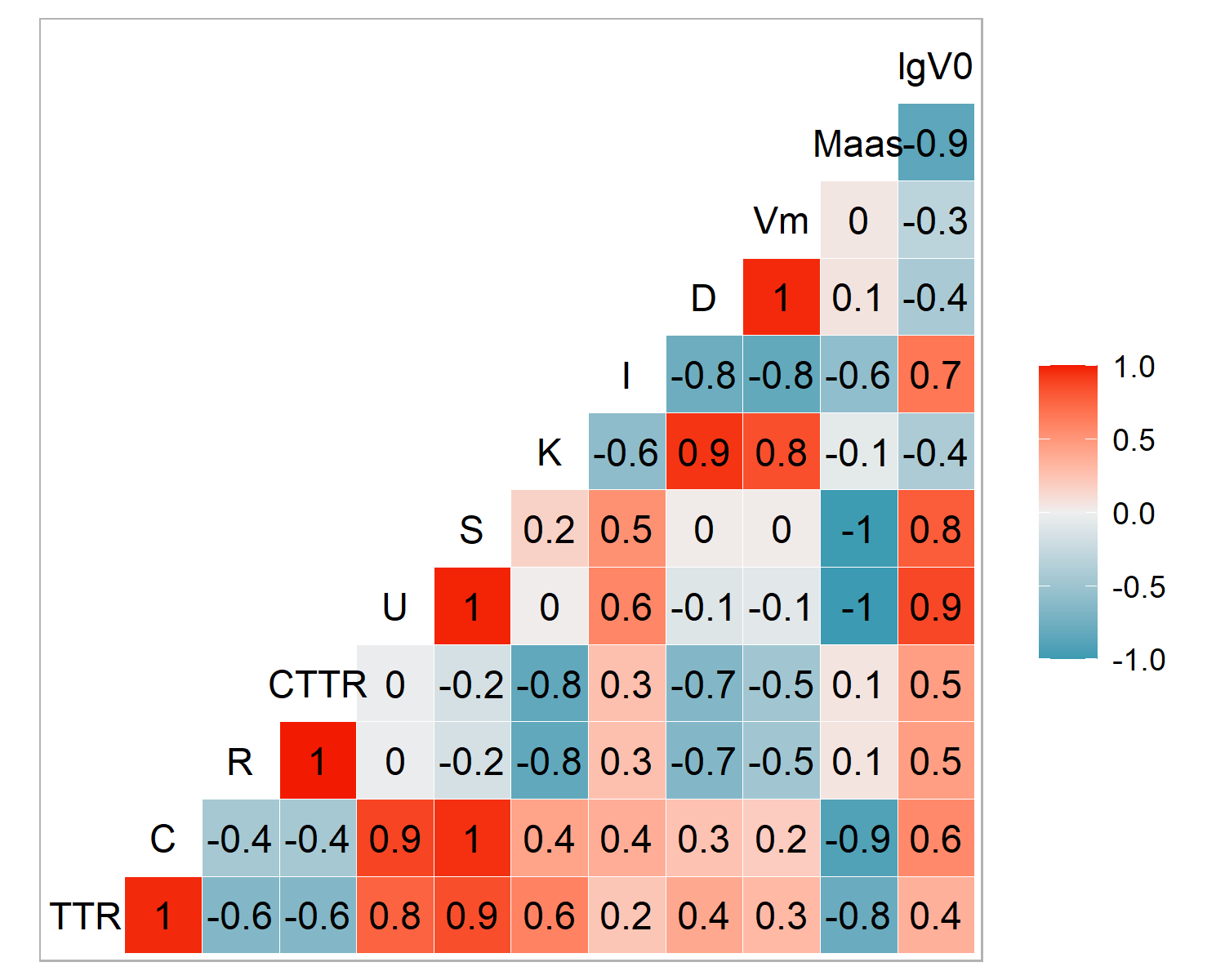
mineln\_dfm %>%  
 textstat\_lexdiv(measure = "all")  
#> document TTR C R CTTR U  
#> 1 antall\_jozsef\_1990 0.6465054 0.9490329 46.65601 32.99078 72.92298  
#> 2 bajnai\_gordon\_2009 0.7283044 0.9566410 28.18830 19.93214 73.23764  
#> 3 boross\_peter\_1993 0.7209677 0.9565427 31.09369 21.98656 75.23509  
#> 4 gyurcsany\_ferenc\_2005 0.5563859 0.9301449 36.97351 26.14422 52.17985  
#> 5 horn\_gyula\_1994 0.7142122 0.9555469 31.47399 22.25547 73.97126  
#> 6 medgyessy\_peter\_2002 0.7110912 0.9514261 23.77641 16.81246 62.75886  
#> 7 orban\_viktor\_1995 0.7224880 0.9574810 33.02964 23.35548 78.08616  
#> 8 orban\_viktor\_2018 0.7529538 0.9584932 22.97435 16.24532 71.52920  
#> S K I D Vm Maas lgV0  
#> 1 0.9601534 11.21473 419.0858 0.0009296389 0.02871363 0.1171029 11.191310  
#> 2 0.9616371 16.40817 459.3906 0.0009739104 0.02691146 0.1168511 10.429595  
#> 3 0.9624951 18.51659 355.0407 0.0013147316 0.03325578 0.1152895 10.725349  
#> 4 0.9440104 11.86397 291.9314 0.0009601654 0.02791768 0.1384359 9.233313  
#> 5 0.9618001 12.60022 571.6996 0.0007454724 0.02321726 0.1162702 10.656921  
#> 6 0.9553276 26.48161 251.3022 0.0017552766 0.03728672 0.1262300 9.420533  
#> 7 0.9637924 16.50145 400.1580 0.0011722374 0.03143078 0.1131652 11.019123  
#> 8 0.9610435 26.17792 313.3935 0.0015453380 0.03451461 0.1182383 9.980932  
#> lgeV0  
#> 1 25.76894  
#> 2 24.01503  
#> 3 24.69603  
#> 4 21.26049  
#> 5 24.53847  
#> 6 21.69158  
#> 7 25.37247  
#> 8 22.98195

Ha pedig arra vagyunk kíváncsiak, hogy a kapott értékek hogyan viszonyulnak egymáshoz, azt a cor() függvény segítésével számolhatjuk ki.

div\_df <- mineln\_dfm %>%  
 textstat\_lexdiv(measure = "all")  
  
cor(div\_df[, 2:13])  
#> TTR C R CTTR U S  
#> TTR 1.0000000 0.9709934 -0.64927227 -0.64927227 0.75866675 0.85148414  
#> C 0.9709934 1.0000000 -0.44988787 -0.44988787 0.88841537 0.95162677  
#> R -0.6492723 -0.4498879 1.00000000 1.00000000 -0.02424963 -0.16264543  
#> CTTR -0.6492723 -0.4498879 1.00000000 1.00000000 -0.02424963 -0.16264543  
#> U 0.7586668 0.8884154 -0.02424963 -0.02424963 1.00000000 0.98430655  
#> S 0.8514841 0.9516268 -0.16264543 -0.16264543 0.98430655 1.00000000  
#> K 0.5991452 0.4294223 -0.83367217 -0.83367217 0.01275208 0.16142157  
#> I 0.2327750 0.3698611 0.26241321 0.26241321 0.58486310 0.52740696  
#> D 0.4013304 0.2514823 -0.65214634 -0.65214634 -0.10697212 0.01957486  
#> Vm 0.3075452 0.1968097 -0.47355632 -0.47355632 -0.07691131 0.01930842  
#> Maas -0.7882881 -0.9112332 0.05759202 0.05759202 -0.99701921 -0.99354677  
#> lgV0 0.3571139 0.5682834 0.45841359 0.45841359 0.87598404 0.79330142  
#> K I D Vm Maas lgV0  
#> TTR 0.59914525 0.2327750 0.40133039 0.30754517 -0.78828811 0.3571139  
#> C 0.42942231 0.3698611 0.25148227 0.19680971 -0.91123321 0.5682834  
#> R -0.83367217 0.2624132 -0.65214634 -0.47355632 0.05759202 0.4584136  
#> CTTR -0.83367217 0.2624132 -0.65214634 -0.47355632 0.05759202 0.4584136  
#> U 0.01275208 0.5848631 -0.10697212 -0.07691131 -0.99701921 0.8759840  
#> S 0.16142157 0.5274070 0.01957486 0.01930842 -0.99354677 0.7933014  
#> K 1.00000000 -0.5873634 0.94208077 0.84481216 -0.06058855 -0.4082192  
#> I -0.58736335 1.0000000 -0.77504336 -0.82248673 -0.56958764 0.6634351  
#> D 0.94208077 -0.7750434 1.00000000 0.96943443 0.06563582 -0.4269281  
#> Vm 0.84481216 -0.8224867 0.96943443 1.00000000 0.04532617 -0.3156220  
#> Maas -0.06058855 -0.5695876 0.06563582 0.04532617 1.00000000 -0.8557245  
#> lgV0 -0.40821917 0.6634351 -0.42692805 -0.31562199 -0.85572445 1.0000000

A kapott értékeket a ggcorr() függvény segítségével ábrázolhatjuk is. Ha a függvény argumentumában a label = TRUE szerepel, a kapott ábrán a kiszámított értékek is szerepelnek.

ggcorr(div\_df[, 2:13], label = TRUE)



Ezt követően azt is megvizsgálhatjuk, hogy a korpusz szövegei mennyire könnyen olvashatóak. Ehhez a Flesch.Kincaid pontszámot használjuk, ami a szavak és mondatok hossza alapján határozza meg a szöveg olvashatóságát. Ehhez textstat\_readability() függvényt használjuk, mely a korpuszunkat elemzi.

corpus\_mineln %>%  
 textstat\_readability(measure = "Flesch.Kincaid")  
#> document Flesch.Kincaid  
#> 1 antall\_jozsef\_1990 16.48512  
#> 2 bajnai\_gordon\_2009 10.92243  
#> 3 boross\_peter\_1993 15.40159  
#> 4 gyurcsany\_ferenc\_2005 13.55911  
#> 5 horn\_gyula\_1994 13.77918  
#> 6 medgyessy\_peter\_2002 15.81893  
#> 7 orban\_viktor\_1995 13.04284  
#> 8 orban\_viktor\_2018 11.39180

Ezután a kiszámított értékkel kiegészítjük a korpuszt.

docvars(corpus\_mineln, "f\_k") <- textstat\_readability(corpus\_mineln, measure = "Flesch.Kincaid")[,   
 2]  
  
docvars(corpus\_mineln)  
#> mineln wing year f\_k  
#> 1 antall\_jozsef jobb 1990 16.48512  
#> 2 bajnai\_gordon bal 2009 10.92243  
#> 3 boross\_peter jobb 1993 15.40159  
#> 4 gyurcsany\_ferenc bal 2005 13.55911  
#> 5 horn\_gyula bal 1994 13.77918  
#> 6 medgyessy\_peter bal 2002 15.81893  
#> 7 orban\_viktor jobb 1995 13.04284  
#> 8 orban\_viktor jobb 2018 11.39180

docvars(corpus\_mineln, "f\_k") <- textstat\_readability(corpus\_mineln, measure = "Flesch.Kincaid")[,   
 2]  
  
docvars(corpus\_mineln)  
#> mineln wing year f\_k  
#> 1 antall\_jozsef jobb 1990 16.48512  
#> 2 bajnai\_gordon bal 2009 10.92243  
#> 3 boross\_peter jobb 1993 15.40159  
#> 4 gyurcsany\_ferenc bal 2005 13.55911  
#> 5 horn\_gyula bal 1994 13.77918  
#> 6 medgyessy\_peter bal 2002 15.81893  
#> 7 orban\_viktor jobb 1995 13.04284  
#> 8 orban\_viktor jobb 2018 11.39180

Majd a ggplot2 segítségével vizualizálhatjuk az eredményt. Ehhez az olvashatósági pontszámmal kiegészített korpuszból data fram-et alakítunk ki, majd beállítjuk az ábrázolás paramétereit. Azaz, hogy a két tengelyen az év illetve az olvashatósági pontszám szerepeljen, a színezés különböztesse meg a jobb és a bal oldalt, az egyes dokumentumokat ponttal jelöljük, a jobb és bal oldali beszédeket vonallal kötjük össze, az ábrára fekete színnel felíratjuk a miniszterelnökök nevét, valamint, hogy az x tengely beosztása az egyes beszédek dátumához igazodjon. A theme\_minimal() függvénnyel pedig azt határozzuk meg, hogy mindez fehér hátteret kapjon.

corpus\_df <- docvars(corpus\_mineln)

ggplot(corpus\_df, aes(year, f\_k, color = wing)) +  
 geom\_point(size = 2) +  
 geom\_line(aes(linetype = wing), size = 1) +  
 geom\_text(aes(label = mineln), color = "black", nudge\_y = 0.15) +  
 scale\_x\_continuous(breaks = corpus\_df$year) +  
 theme\_minimal()

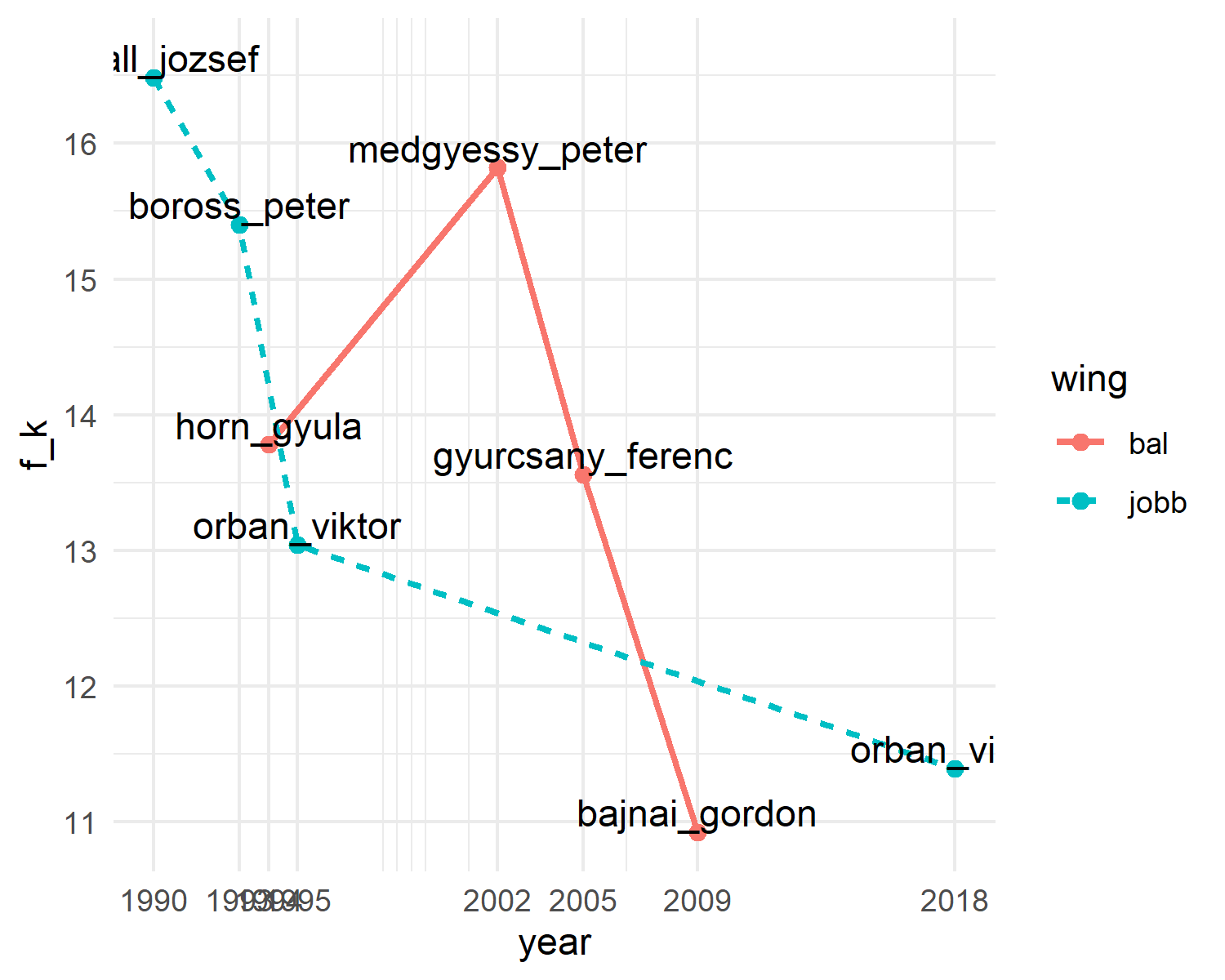


Figure 5.1: Az olvashatósági index alakulása

## 5.4 Összehasonlítás[[13]](#footnote-91)

A fentiekben láthattuk az eltéréseket a jobb és bal oldali beszédeken belül, sőt ugyanahhoz a miniszterelnökhöz tartozó két beszéd között is. A következőkben textstat\_dist() és textstat\_simil() függvények segítségével megvizsgáljuk, valójában mennyire hasonlítanak vagy különböznek ezek a beszédek. Mindkét függvény bemenete dmf, melyből először egy súlyozott dfm-et készítünk, majd elvégezzük az összehasonlítást először a jaccard-féle hasonlóság alapján.

mineln\_dfm %>%  
 dfm\_weight("prop") %>%  
 textstat\_simil(margin = "documents", method = "jaccard")  
#> textstat\_simil object; method = "jaccard"  
#> antall\_jozsef\_1990 bajnai\_gordon\_2009 boross\_peter\_1993  
#> antall\_jozsef\_1990 1.0000 0.0559 0.1011  
#> bajnai\_gordon\_2009 0.0559 1.0000 0.0564  
#> boross\_peter\_1993 0.1011 0.0564 1.0000  
#> gyurcsany\_ferenc\_2005 0.0798 0.0850 0.0604  
#> horn\_gyula\_1994 0.0694 0.0592 0.0613  
#> medgyessy\_peter\_2002 0.0404 0.0690 0.0473  
#> orban\_viktor\_1995 0.0778 0.0626 0.0631  
#> orban\_viktor\_2018 0.0362 0.0617 0.0401  
#> gyurcsany\_ferenc\_2005 horn\_gyula\_1994  
#> antall\_jozsef\_1990 0.0798 0.0694  
#> bajnai\_gordon\_2009 0.0850 0.0592  
#> boross\_peter\_1993 0.0604 0.0613  
#> gyurcsany\_ferenc\_2005 1.0000 0.0683  
#> horn\_gyula\_1994 0.0683 1.0000  
#> medgyessy\_peter\_2002 0.0684 0.0587  
#> orban\_viktor\_1995 0.0734 0.0621  
#> orban\_viktor\_2018 0.0503 0.0494  
#> medgyessy\_peter\_2002 orban\_viktor\_1995 orban\_viktor\_2018  
#> antall\_jozsef\_1990 0.0404 0.0778 0.0362  
#> bajnai\_gordon\_2009 0.0690 0.0626 0.0617  
#> boross\_peter\_1993 0.0473 0.0631 0.0401  
#> gyurcsany\_ferenc\_2005 0.0684 0.0734 0.0503  
#> horn\_gyula\_1994 0.0587 0.0621 0.0494  
#> medgyessy\_peter\_2002 1.0000 0.0650 0.0504  
#> orban\_viktor\_1995 0.0650 1.0000 0.0583  
#> orban\_viktor\_2018 0.0504 0.0583 1.0000

Majd a textstat\_dist() függvény segítségével kiszámoljuk a dokumentumok egymástól való különbözőségét.

mineln\_dfm %>%  
 textstat\_dist(margin = "documents", method = "euclidean")  
#> textstat\_dist object; method = "euclidean"  
#> antall\_jozsef\_1990 bajnai\_gordon\_2009 boross\_peter\_1993  
#> antall\_jozsef\_1990 0 162.8 134.1  
#> bajnai\_gordon\_2009 163 0 84.6  
#> boross\_peter\_1993 134 84.6 0  
#> gyurcsany\_ferenc\_2005 186 137.8 149.7  
#> horn\_gyula\_1994 164 80.1 88.0  
#> medgyessy\_peter\_2002 160 68.1 81.8  
#> orban\_viktor\_1995 139 84.7 79.7  
#> orban\_viktor\_2018 167 67.3 85.4  
#> gyurcsany\_ferenc\_2005 horn\_gyula\_1994  
#> antall\_jozsef\_1990 186 163.6  
#> bajnai\_gordon\_2009 138 80.1  
#> boross\_peter\_1993 150 88.0  
#> gyurcsany\_ferenc\_2005 0 147.0  
#> horn\_gyula\_1994 147 0  
#> medgyessy\_peter\_2002 143 75.9  
#> orban\_viktor\_1995 147 89.6  
#> orban\_viktor\_2018 148 74.8  
#> medgyessy\_peter\_2002 orban\_viktor\_1995 orban\_viktor\_2018  
#> antall\_jozsef\_1990 160.2 139.2 167.4  
#> bajnai\_gordon\_2009 68.1 84.7 67.3  
#> boross\_peter\_1993 81.8 79.7 85.4  
#> gyurcsany\_ferenc\_2005 142.6 146.9 147.9  
#> horn\_gyula\_1994 75.9 89.6 74.8  
#> medgyessy\_peter\_2002 0 77.5 60.7  
#> orban\_viktor\_1995 77.5 0 83.6  
#> orban\_viktor\_2018 60.7 83.6 0

Ezután vizualizálhatjuk is a dokumentumok egymástól való távolságát egy olyan dendogram segítségével, amely megmutatja nekünk a lehetséges dokumentumpárokat.

dist <- mineln\_dfm %>%  
 textstat\_dist(margin = "documents", method = "euclidean")

plot(hclust(as.dist(dist)))

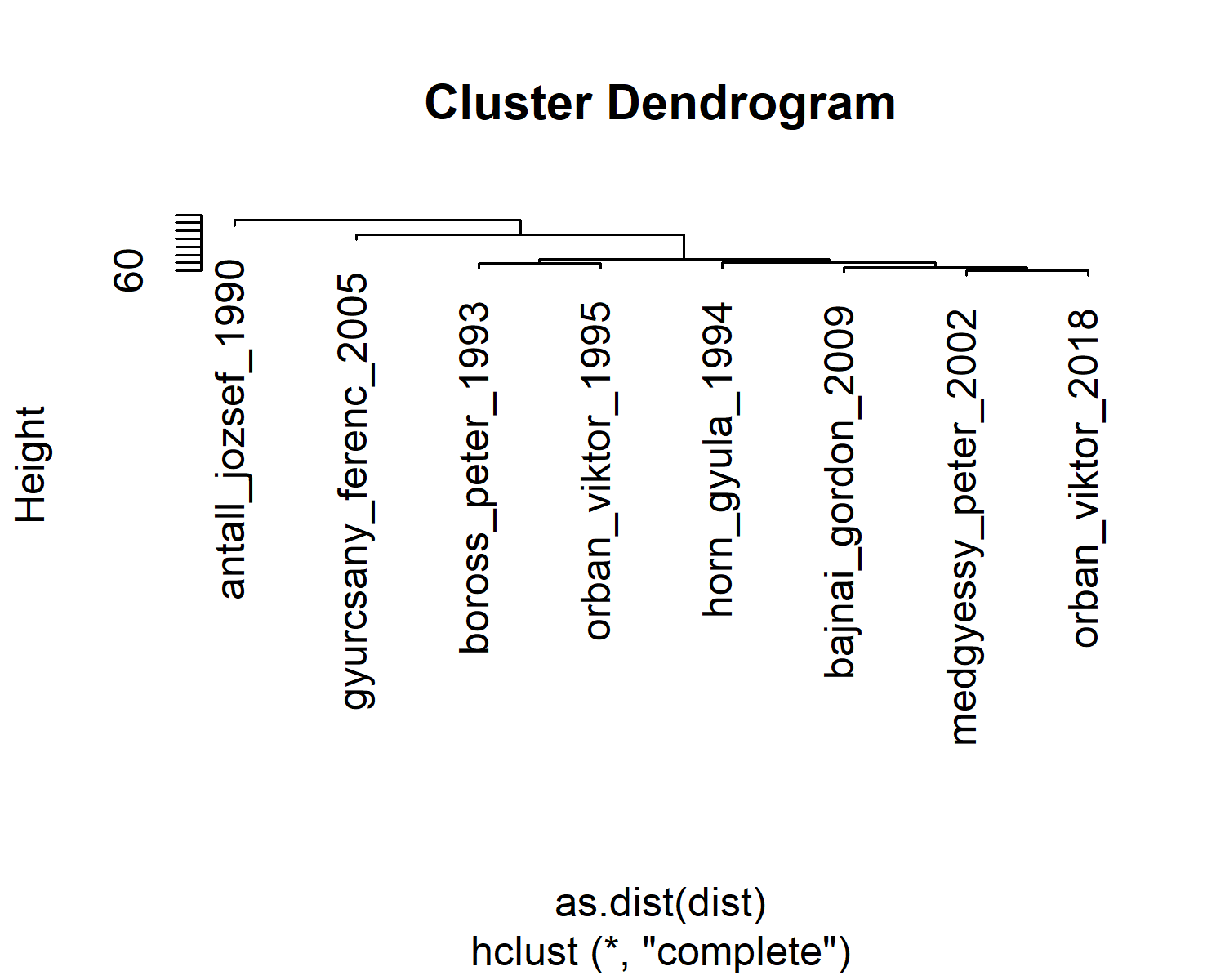


Figure 5.2: A dokumentumok csoportosítása a távolságuk alapján

mineln\_dfm %>%  
 textstat\_simil(y = mineln\_dfm[, c("kormány")], margin = "features", method = "correlation") %>%  
 head(n = 10)  
#> kormány  
#> elnök -0.1185449  
#> tisztelt -0.5401371  
#> országgyulés 0.8082763  
#> hölgyeim -0.3580776  
#> uraim -0.3580776  
#> honfitársaim 0.8648335  
#> ünnepi 0.8737759  
#> pillanatban 0.8737759  
#> állok 0.6864778  
#> magyar 0.7530170

Arra is van lehetőségünk, hogy a két alkorpuszt hasonlítsuk össze egymással. Ehhez a textstat\_keyness() függvényt használjuk, melynek a bemenete a dfm. A függvény argumentumában a target = után kell megadni, hogy mely alkorpusz a viszonyítási alap. Az összehasonlítás eredményét a ’textplot\_keyness()` függvény segítségével ábrázolhatjuk, ami megjeleníti a két alkorpusz leggyakoribb kifejezéseit.

dfm <- dfm(  
 corpus\_mineln,  
 groups = "wing",   
 remove = custom\_stopwords,  
 remove\_punct = TRUE  
)  
  
result\_keyness <- textstat\_keyness(dfm, target = "jobb")

textplot\_keyness(result\_keyness)

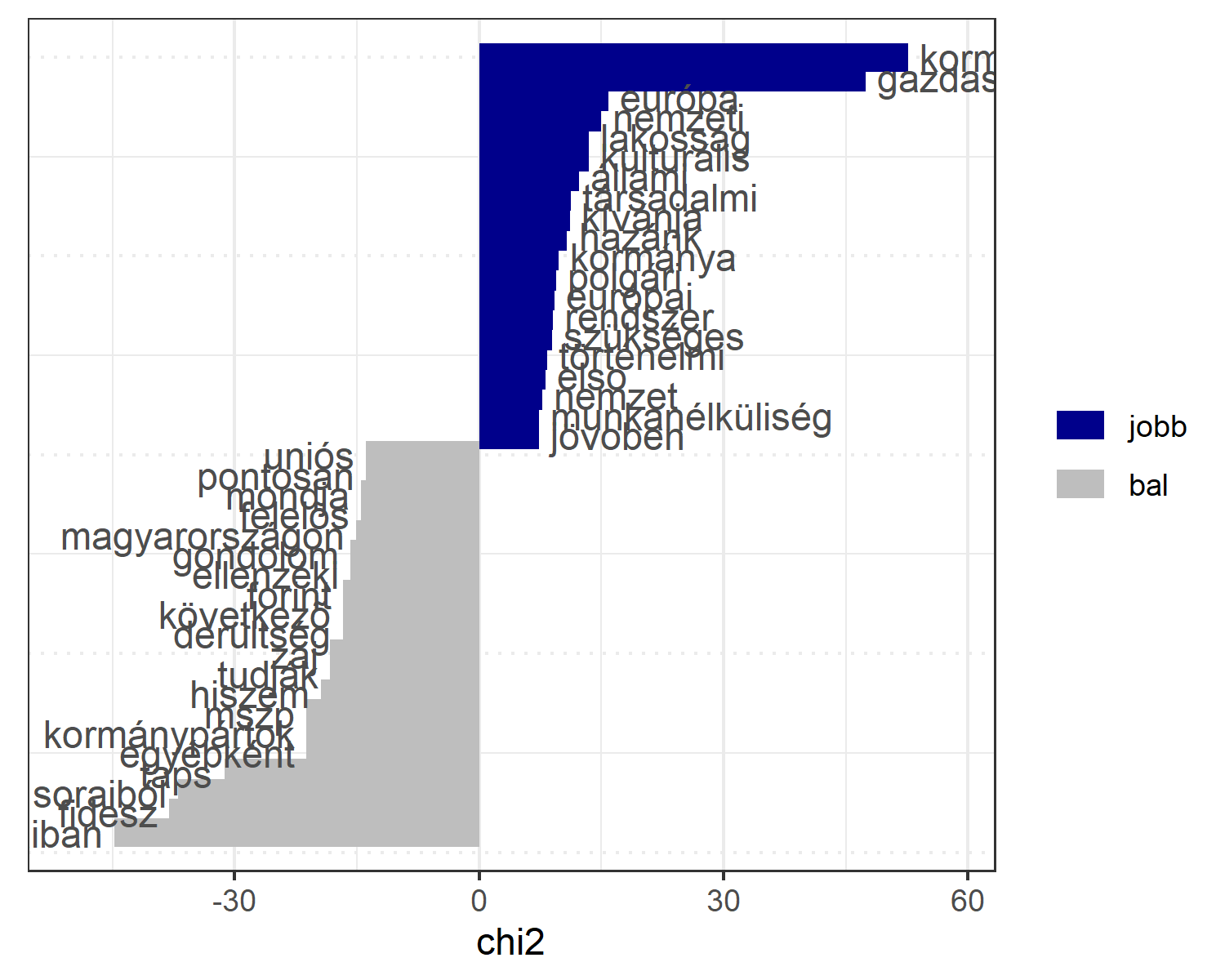


Figure 5.3: A korpuszok legfontosabb kifejezései

Ha az egyes miniszterelnökök beszédeinek leggyakoribb kifejezéseit szeretnénk összehasonlítani, azt a textstat\_frequency() függvény segítségével tehetjük meg, melynek bemenete a megtisztított és súlyozott dfm. Az összehasonlítás eredményét pedig a ggplot2 segítségével ábrázolhatjuk is.

dfm\_weight <- corpus\_mineln %>%  
 dfm(  
 remove = custom\_stopwords,   
 tolower = TRUE,   
 remove\_punct = TRUE,   
 stem = TRUE,   
 remove\_symbols = TRUE,   
 remove\_numbers = TRUE  
 ) %>%  
 dfm\_weight(scheme = "prop")  
  
freq\_weight <- textstat\_frequency(dfm\_weight, n = 5, groups = "mineln")

ggplot(data = freq\_weight, aes(x = nrow(freq\_weight):1, y = frequency)) +  
 geom\_point() +  
 facet\_wrap(~ group, scales = "free") +  
 coord\_flip() +  
 scale\_x\_continuous(  
 breaks = nrow(freq\_weight):1,  
 labels = freq\_weight$feature  
 ) +  
 labs(  
 x = NULL,   
 y = "Relative frequency"  
 )

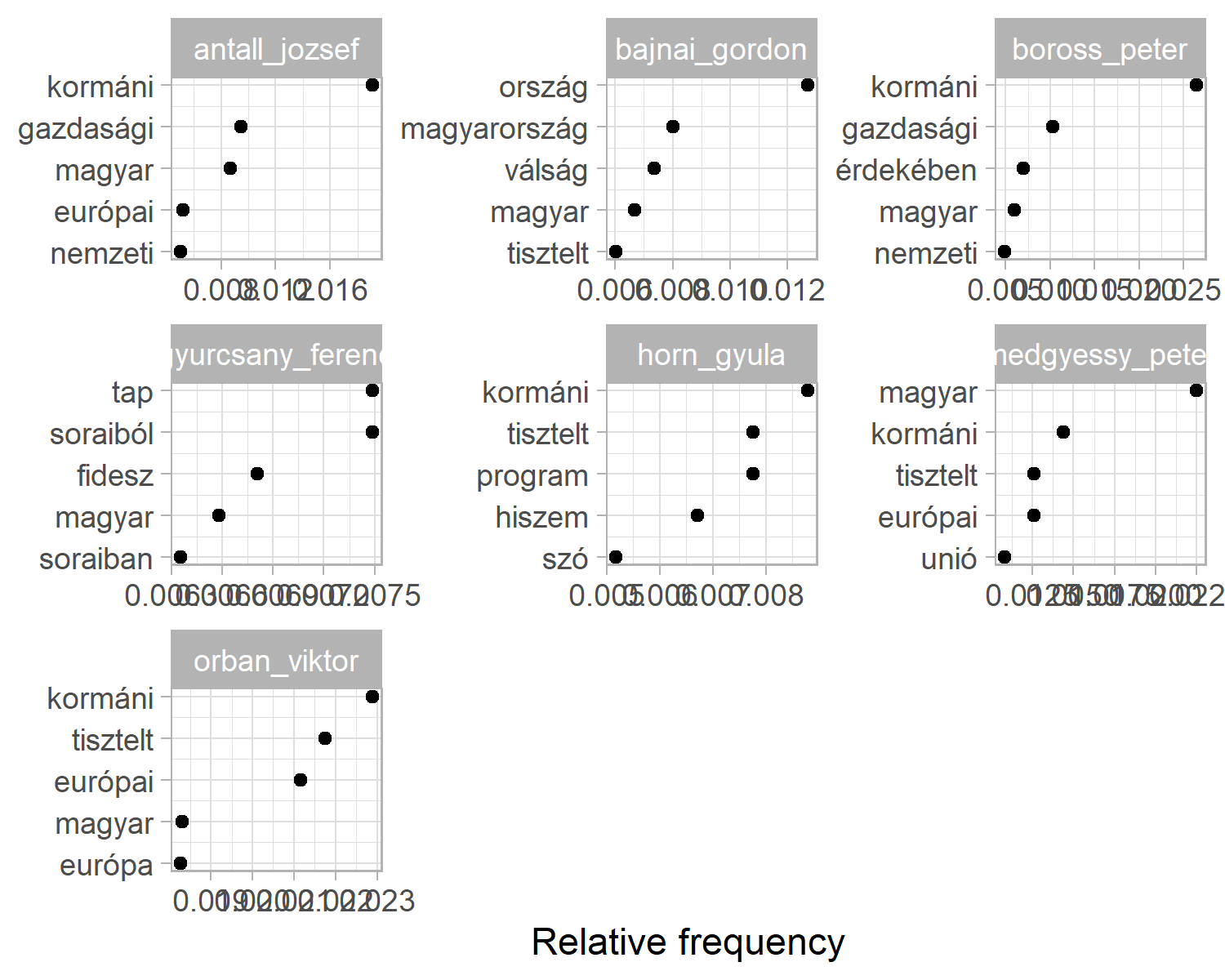


Figure 5.4: Leggyakoribb kifejezések a miniszterelnöki beszédekben

## 5.5 Kifejezések kontextusba helyezése

Arra is lehetőségünk van, hogy egyes kulcszavakat a korpuszon belül szövegkörnyezetükben vizsgáljunk meg. Ehhez a kwic() függvényt használjuk, az argumentumok között a pattern = kifejezés után megadva azt a szót, amelyet vizsgálni szeretnénk, a ‘window =’ után pedig megadhatjuk hogy az adott szó hány szavas környezetére vagyunk kíváncsiak.

kwic(corpus\_mineln, pattern = "válság\*", valuetype = "glob", window = 3, case\_insensitive = TRUE) %>%  
 head(5)  
#>   
#> [antall\_jozsef\_1990, 1167] Átfogó és mély | válságba |  
#> [antall\_jozsef\_1990, 1283] kell hárítanunk a | válságot |  
#> [antall\_jozsef\_1990, 2772] és a lakásgazdálkodás | válságos |  
#> [antall\_jozsef\_1990, 5226] gazdaság egészét juttatta | válságba |  
#> [antall\_jozsef\_1990, 5286] gazdaság reménytelenül eladósodott | válsággócai |  
#>   
#> süllyedtünk a nyolcvanas  
#> , de csakis   
#> helyzetbe került.   
#> , és amellyel   
#> ellen. A

# 6 Szótáralapú elemzések, érzelem-elemzés

A szentimentelemzés a számítógépes nyelvészet részterülete, melynek célja az egyes szövegek tartalmából kinyerni azokat az információkat, amelyek értékelés fejeznek ki, azaz szentimentértékkel bírnak.[[14]](#footnote-98) A szentimentelemzés a szövegeket három szinten osztályozza. A legáltalánosabb a dokumentumszintű osztályozás, amikor egy hosszabb szövegegység egészét vizsgáljuk, míg a mondatszintű osztályozásnál a vizsgálat alapegysége a mondat. A legrészletesebb adatokat akkor nyerjük, amikor az elemzést target-szinten végezzük, azaz meghatározzuk azt is, hogy egy-egy érzelem a szövegen belül mire vonatkozik. Mindhárom szinten azonos a feladat, egyrészt meg kell állapítani, hogy az adott egységben van-e értékelés, vélemény vagy érzelem és ha igen, akkor pedig meg kell határozni, hogy milyen azok érzelmi tartalma. A pozitív-negatív-semleges skálán mozgó szentimentelemzéshez mellett az elmúlt két évtizedben jelentős lépések történetek a szövegek emóciótartalmának automatikus vizsgálatára is. A módszer hasonló a szentimentelemzéshez, tartalmilag azonban más skálán mozog. Az emócióelemzés esetén ugyanis, nem csak azt kell meghatározni, hogy egy kifejezés pozitív bvagy negatív tölettel rendelkezik, hanem azt is, hogy milyen érzelmet (öröm, bánat, undor, stb) hordoz. A szótár alapú szentiment- vagy emócióelemzés alapja az az egy egyszerű ötlet, hogy ha tudjuk, hogy egyes szavak milyen érzelmeket, érzéseket hordoznak, akkor ezeket a szavakat egy szövegben megszámolva képet kaphatunk az adott dokumentum érzelmi tartalmáról.Mivel a szótár alapú elemzés az adott szentiment kategórián belüli kulcsszavak gyakoriságán alapul, ezért van aki nem tekinti statisztikai elemzésnek (lásd például [Young and Soroka](#ref-young2012affective) ([2012](#ref-young2012affective))). A tágabb kvantitatív szövegelemzési kontextusban az osztályozáson (classification) belül a felügyelt módszerekhez hasonlóan itt is ismert kategóriákkal dolgozunk, azaz előre meghatározzuk, hogy egy-egy adott szó pozitív vagy negatív szentimentértékű, vagy továbbmenve, milyen érzelmet hordoz csak egyszerűbb módszertannal ([Grimmer and Stewart 2013](#ref-grimmer2013text)).A kulcsszavakra építés miatt a módszer a kvalitatív és kvantitatív kutatási vonalak találkozásának is tekinthető, hiszen egy-egy szónak az érzelmi töltete nem mindig ítélhető meg objektíven. Mint minden módszer esetében, itt is kiemelten fontos hogy ellenőrízzük hogy a használt szótár kategóriák és kulcsszavak fedik-e a valóságot. Más szavakkal: *validate, validate, validate*.

**A módszer előnyei:**

* Tökéletesen megbízható: a számításoknak nincsen probabilisztikus (azaz valószínűségre épülő) eleme, mint például a Support Vector alapú osztályozásnak, illetve az emberi szövegkódolásnál előforduló problémákat (például, hogy két kódoló, vagy ugyanazon kodóló két különböző időpontban nem azonosan értékeli ugyanazt a kifejezést) is elkerüljük így.
* Képesek vagyunk vele mérni a szöveg látens dimenzióit.
* Széles körben alkalmazható, egyszerűen számolható. A politikatudományon és számítógépes nyelvészeten belül nagyon sok kész szótár elérhető, amelyek különböző módszerekkel készültek és különböző területet fednek le (pl.: populizmus, pártprogramok policy tartalma, érzelmek, gazdasági tartalom.)[[15]](#footnote-99)
* Relatíve könnyen adaptálható egyik nyelvi környezetből másikba, bár szótárfordítások esetén külön hangsúlyt kell fektetni a validálásra.

**A módszer lehetéges hátrányai:**

* A szótár hatékonysága és validitása azon múlik hogy mennyire egyezik a szótár és a vizsgálni kívánt dokumentum területe. Nem mindegy például, hogy a szótárunkkal a gazdasági bizonytalanságot szeretnék tőzsdei jelentések alapján vizsgálni vagy nézők filmekre adott értékeléseit. Léteznek általános szentimentszótárak, ezek hatékonysága azonban általában alulmúlja a terület-specifikus szótárakét. (Nem mindegy például, hogy valaminek )
* A terület-specifikus szótár építése egy kvalitatív folyamat (lsd. a lábjegyzetben), éppen ezért idő és emberi erőforrás igényes.
* A szózsák alapú elemzéseknél a kontextus elvész. Gondoljunk például a tagadásra: a *“nem vagyok boldog”* kifejezés esetén egy általános szentiment szótár a tagadás miatt félreosztályozná a mondat érzelmi töltését, hiszen a boldog szó önmagában a pozitív kategóriába tartozik. Természetesen az automatikus tagadás-kezelésre is vannak lehetőségek, de a kérdés komplexitása miatt ezek bemutatásától most eltekintünk.

A legnagyobb méretű általános szentimentszótár az angol nyelvű SentiWordNet (SWN), ami kb. 150 000 szót tartalmaz, amelyek mindegyike a három szentimentérték - pozitív, negatív, semleges - közül kapott egyet.[[16]](#footnote-100)([Baccianella, Esuli, and Sebastiani 2010](#X46e58594640e59787c4703545ad8daea29317b9))

Az R-ben végzett szentimentelemzés során, angol nyelvű szövegekhez több beépített általános szentimentszótár is a rendelkezésünkre áll.[[17]](#footnote-102) A teljesség igénye nélkül említhetjük az AFINN[[18]](#footnote-104), a bing[[19]](#footnote-106) és a az nrc[[20]](#footnote-108) szótárakat. Az elemzés sikere több faktortól is függ. Fontos hogy a korpuszban lévő dokumentumokat körültekintően tisztítsuk meg az elemzés elején (lásd a 4. fejezetet a szövegelőkészítésről). A következő lépésben meg kell bizonyosodnunk arról, hogy a kiválasztott szentiment szótár alkalmazható a korpuszunkra. Amennyiben nem találunk alkalmas szótárt, akkor a saját szótár validálására kell figyelni. A negyedik fejezetben leírtak itt is érvényesek, érdemes a dokumentum-kifejezés mátrixot súlyozni valamilyen módon.

## 6.1 Szótárak az R-ben

A szótár alapú elemzéshez a quanteda csomagot fogjuk használni, illetve a 3. fejezetben már megismert readr, stringr, dplyr tidyverse csomagokat.[[21]](#footnote-110)

library(readr)  
library(stringr)  
library(dplyr)  
library(ggplot2)  
library(quanteda)

Mielőtt a két esettanulmányt bemutatnánk, vizsgáljuk meg hogy hogyan néz ki egy szentiment szótár az R-ben. A szótárt kézzel úgy tudjuk létrehozni, hogy egy listán belül létrehozzuk karaktervektorként a kategóriákat és a kulcsszavakat és ezt a listát a quanteda dictionary függvényével eltároljuk.

szentiment\_szotar <- dictionary(  
 list(  
 pozitiv = c("jó", "boldog", "öröm"),  
 negativ = c("rossz", "szomorú", "lehangoló")  
 )  
 )  
  
szentiment\_szotar  
#> Dictionary object with 2 key entries.  
#> - [pozitiv]:  
#> - jó, boldog, öröm  
#> - [negativ]:  
#> - rossz, szomorú, lehangoló

A quanteda, quanteda.corpora és tidytext R csomagok több széles körben használt szentiment szótárat tartalmaznak, így nem kell kézzel replikálni minden egyes szótárat amit használni szeretnénk.

A szentiment elemzési munkafolyamat amit a részfejezetben bemutatunk a következő lépésekből áll:

1. dokumentumok betöltése
2. szöveg előkészítése
3. a korpusz létrehozása
4. dokumentum-kifejezés mátrix
5. szótár betöltése
6. a dokumentum-kifejezés mátrix szűrése a szótárban lévő kulcsszavakkal
7. az eredmény vizualizálása, további felhasználása

A fejezetben két különböző korpuszt fogunk elemezni: a 2006-os Magyar Nemzet címlapjainak egy 252 cikkből álló mintájának szentimentjét vizsgáljuk egy magyar szentiment szótárral.[[22]](#footnote-112) A második korpusz a Magyar Nemzeti Bank angol nyelvű sajtóközleményeiből áll, amin bemutatjuk egy széles körben használt gazdasági szótár használatát.[[23]](#footnote-114)

## 6.2 Magyar Nemzet cikkek

mn\_minta <- read\_csv("data/magyar\_nemzet\_small.csv")  
  
summary(mn\_minta)  
#> doc\_id text doc\_date   
#> Min. : 1.0 Length:2834 Min. :2006-01-02   
#> 1st Qu.: 709.2 Class :character 1st Qu.:2006-03-29   
#> Median :1417.5 Mode :character Median :2006-06-28   
#> Mean :1417.5 Mean :2006-06-28   
#> 3rd Qu.:2125.8 3rd Qu.:2006-09-26   
#> Max. :2834.0 Max. :2006-12-29

A read\_csv() segítségével beolvassuk a Magyar Nemzet adatbázis egy kisebb részét, ami az esetünkben a 2006-os címlapokon szereplő hírek . A summary(), ahogy a neve is mutatja, egy gyors áttekintést nyújt a betöltött adatbázisról. Látjuk, hogy 2834 sorból (megfigyelés) és 3 oszlopból (változó) áll. Az első ránézésre látszik hogy a text változónk tartalmazza a szövegeket, és hogy tisztításra szorulnak.

Az első szöveget megnézve látjuk, hogy a standard előkészítési lépések mellett a sortörést (\n) is ki kell törölnünk.

mn\_minta$text[1]  
#> [1] "Hat fovárosi képviselo öt percnél is kevesebbet beszélt egy év alatt a közgyulésben.\n\n\n\n\n\n\n\n\n\n\n"

Habár a quanteda is lehetőséget ad néhány előkészítő lépésre, érdemes ezt olyan céleszközzel tenni ami nagyobb rugalmasságot ad a kezünkbe. Mi erre a célra a stringr csomagot használjuk. Első lépésben kitöröljük a sortöréseket (\n), a központozást, számokat, kisbetűsítünk minden szót. Előfordulhat hogy (számunkra nehezen látható) extra szóközök maradnak a szövegben. Ezeket az str\_squish()-el tüntetjük el. A szöveg eleji és végi extra szóközöket (ún. leading vagy trailing white space) az str\_trim() vágja le.

mn\_tiszta <- mn\_minta %>%  
 mutate(  
 text = str\_remove\_all(string = text, pattern = "\n"),  
 text = str\_remove\_all(string = text, pattern = "[:punct:]"),  
 text = str\_remove\_all(string = text, pattern = "[:digit:]"),  
 text = str\_to\_lower(text),  
 text = str\_trim(text),  
 text = str\_squish(text)  
 )

A szöveg sokkal jobban néz ki, habár észrevehetjük hogy maradhattak benne problémás részek, főleg a sortörés miatt, ami sajnos hol egyes szavak közepén van (a jobbik eset), vagy pedig pont szóhatáron, ez esetben a két szó sajnos összevonódik. Az egyszerűség kedvéért feltételezzük hogy ez kellően ritkán fordul elő ahhoz hogy ne befolyásolja az elemzésünk eredményét.

mn\_tiszta$text[1]  
#> [1] "hat fovárosi képviselo öt percnél is kevesebbet beszélt egy év alatt a közgyulésben"

Miután kész a tiszta(bb) szövegünk, kreálunk egy korpuszt a quanteda corpus() fuggvenyevel. A létrehozott corpus objektum a szöveg mellett egyéb dokumentum meta adatokat is tud tárolni (dátum, író, hely, stb.) Ezeket mi is hozzáadhatjuk (erre majd látunk példát nemsokára) illetve amikor létrehozzuk a korpuszt a data frame-ünkből, akkor automatikusan meta adatokként tárolódnak az változóink. Jelen esetben az egyetlen dokumentum változónk az a dátum lesz a szöveg mellett. A korpusz dokumentum változóihoz a docvars() segítségével tudunk hozzáférni.

mn\_corpus <- corpus(mn\_tiszta)  
  
head(docvars(mn\_corpus), 5)  
#> doc\_date  
#> 1 2006-01-02  
#> 2 2006-01-02  
#> 3 2006-01-02  
#> 4 2006-01-02  
#> 5 2006-01-02

A következő lépés a dokumentum-kifejezés mátrix létrehozása a dfm() függvénnyel (ami a *document-feature matrix* rövidítése). Előszőr tokenekre bontjuk a szövegeket a tokens()-el, és aztán ezt a tokenizált szózsákot kapja meg a dfm inputnak. A sornak a végén a létrehozott mátrixunkat TF-IDF módszerrel súlyozzuk a dfm\_tfidf() használatával.

mn\_dfm <- mn\_corpus %>%  
 tokens(what = "word") %>%  
 dfm() %>%  
 dfm\_tfidf()

A cikkek szentimentjét egy magyar szótárral fogjuk becsülni, amit a Társadalomtudományi Kutatóközpont kutatói a Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében készítettek.[[24]](#footnote-118) Két dimenziót tarlamaz (pozitív és negatív), 2614 pozitív és 2654 negatív kulcsszóval. Ez nem számít kirívóan nagynak a szótárak között, mivel az adott kategóriák minél teljesebb lefedése a cél.

poltext\_szotar <- read\_rds("data/poltext\_dict.Rds")

poltext\_szotar  
#> Dictionary object with 2 key entries.  
#> - [positive]:  
#> - abszolút, ad, adakozó, adaptív, adekvát, adócsökkentés, adókedvezmény, adomány, adományoz, adóreform, adottság, adottságú, áfacsökkentés, agilis, agytröszt, áhítatosság, áhított, ajándék, ajándékoz, ajánl [ ... and 2,594 more ]  
#> - [negative]:  
#> - aberrált, abnormális, abszurd, ádázul, addikció, adócsalás, adóelhárítás, adókijátszás, adókikerülés, aggály, aggasztó, aggodalom, aggódás, aggódik, agitátor, agónia, agresszivitás, agresszor, agyoncsépelt, agyonismételt [ ... and 2,634 more ]

Az egyes dokumentumok szentimentjét a dfm\_lookup() becsüli, ahol az előző lépésben létrehozott súlyozott dfm az input és a magyar szentimentszótár a dictionary. Egy gyors pillantás az eredményre és látjuk hogy minden dokumentumhoz készült egy pozitív és egy negatív értéket. A TF-IDF súlyozás miatt nem látunk egész számokat (a súlyozás nélkül a sima szófrekvenciát kapnánk).

mn\_szentiment <- dfm\_lookup(mn\_dfm, dictionary = poltext\_szotar)  
  
head(mn\_szentiment, 5)  
#> Document-feature matrix of: 5 documents, 2 features (40.0% sparse) and 1 docvar.  
#> features  
#> docs positive negative  
#> 1 0 0   
#> 2 0.8375026 10.219219  
#> 3 0 0   
#> 4 21.1044299 4.633902  
#> 5 11.0358129 5.524588

Ahhoz hogy fel tudjuk használni a kapott eredményt, érdemes dokumentumváltozóként eltárolni a korpuszban. Ezt a fent már használt docvars() segítségével tudjuk megtenni, ahol a második argumentumkét az új változó nevét adjuk meg stringként.

docvars(mn\_corpus, "pos") <- as.numeric(mn\_szentiment[, 1])  
docvars(mn\_corpus, "neg") <- as.numeric(mn\_szentiment[, 2])  
  
head(docvars(mn\_corpus), 5)  
#> doc\_date pos neg  
#> 1 2006-01-02 0.0000000 0.000000  
#> 2 2006-01-02 0.8375026 10.219219  
#> 3 2006-01-02 0.0000000 0.000000  
#> 4 2006-01-02 21.1044299 4.633902  
#> 5 2006-01-02 11.0358129 5.524588

Végül a kapott korpuszt a kiszámolt szentiment értékekkel a quanteda-ban lévő convert() fügvénnyel data frame-é alakítjuk. A convert függvény dokumentációját érdemes elolvasni, mert ennek segítségével tudjuk a quanteda-ban elkészült objektumainkat átalakítani úgy, hogy azt más csomagok is tudják használni.

mn\_df <- convert(mn\_corpus, to = "data.frame")  
  
  
summary(mn\_df)  
#> doc\_id text doc\_date pos   
#> Length:2834 Length:2834 Min. :2006-01-02 Min. : 0.000   
#> Class :character Class :character 1st Qu.:2006-03-29 1st Qu.: 0.000   
#> Mode :character Mode :character Median :2006-06-28 Median : 2.421   
#> Mean :2006-06-28 Mean : 4.149   
#> 3rd Qu.:2006-09-26 3rd Qu.: 6.391   
#> Max. :2006-12-29 Max. :38.114   
#> neg   
#> Min. : 0.000   
#> 1st Qu.: 0.000   
#> Median : 1.745   
#> Mean : 3.066   
#> 3rd Qu.: 4.640   
#> Max. :29.346

Mielőtt vizualizálnánk az eredményt érdemes a napi szintre aggregálni a szentimentet és egy nettó értéket kalkulálni.[[25]](#footnote-121)

mn\_df <- mn\_df %>%  
 group\_by(doc\_date) %>%  
 summarise(  
 daily\_pos = sum(pos),  
 daily\_neg = sum(neg),  
 net\_daily = daily\_pos - daily\_neg  
 )

A plot alapján a szentimentekben több kiugrást is talapsztalhatunk. Természetesen messzemenő következtetéseket egy ilyen kis korpusz alapján nem vonhatunk le, de a kiugrásokhoz tartozó cikkek kvalitatív vizsgálatával megállapíthatjuk, hogy az áprilisi kiugrást a választásokhoz kötődő cikkek pozitív hangulata, míg az októberi negatív kilengést az öszödi beszéd nyilvánosságra kerüléséhez köthető cikkek negatív szentimentje okozza.

ggplot(mn\_df, aes(doc\_date, net\_daily)) + geom\_line() + labs(y = "Szentiment", x = NULL,   
 caption = "Adatforrás: https://cap.tk.hu/")

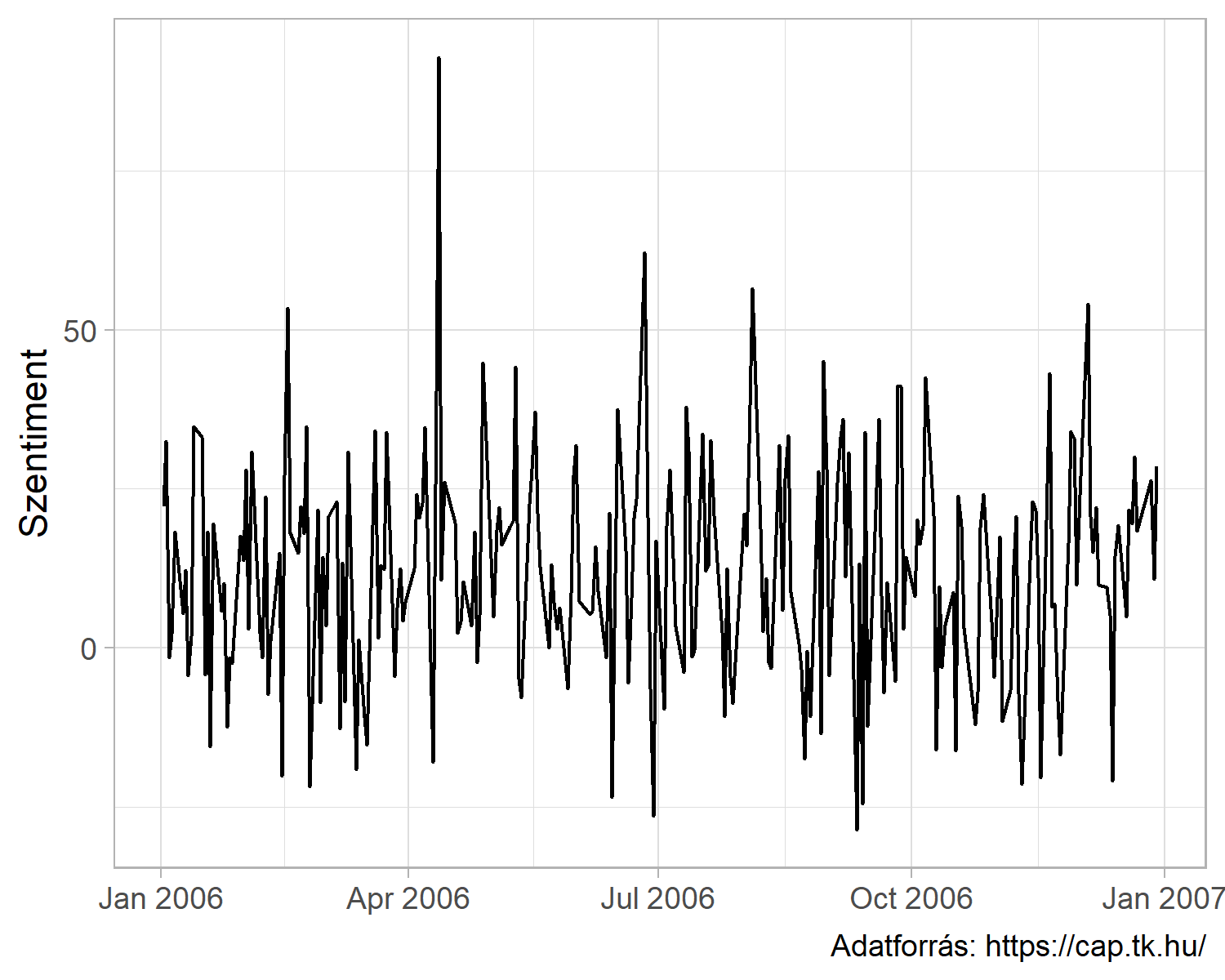


Figure 6.1: Magyar Nemzet címlap szentimentje

## 6.3 MNB sajtóközlemények

A második esettanulmányban a kontextuális szótár elemzést mutatjuk be egy angol nyelvű korpusz és specializált szótár segítségével. A korpusz az MNB kamatdöntéseit kísérő nemzetközi sajtóközleményei és a szótár pedig a [Loughran and McDonald](#ref-loughranWhenLiabilityNot2011) ([2011](#ref-loughranWhenLiabilityNot2011)) pénzügyi szentimentszótár.[[26]](#footnote-124) A szótár a quanteda.dictionaries csomag részeként elérhető, illetve a tankönyv honlapján is megtalálható.

penzugy\_szentiment <- readRDS("data/dictionary\_LoughranMcDonald.Rds")

penzugy\_szentiment  
#> Dictionary object with 9 key entries.  
#> - [NEGATIVE]:  
#> - abandon, abandoned, abandoning, abandonment, abandonments, abandons, abdicated, abdicates, abdicating, abdication, abdications, aberrant, aberration, aberrational, aberrations, abetting, abnormal, abnormalities, abnormality, abnormally [ ... and 2,335 more ]  
#> - [POSITIVE]:  
#> - able, abundance, abundant, acclaimed, accomplish, accomplished, accomplishes, accomplishing, accomplishment, accomplishments, achieve, achieved, achievement, achievements, achieves, achieving, adequately, advancement, advancements, advances [ ... and 334 more ]  
#> - [UNCERTAINTY]:  
#> - abeyance, abeyances, almost, alteration, alterations, ambiguities, ambiguity, ambiguous, anomalies, anomalous, anomalously, anomaly, anticipate, anticipated, anticipates, anticipating, anticipation, anticipations, apparent, apparently [ ... and 277 more ]  
#> - [LITIGIOUS]:  
#> - abovementioned, abrogate, abrogated, abrogates, abrogating, abrogation, abrogations, absolve, absolved, absolves, absolving, accession, accessions, acquirees, acquirors, acquit, acquits, acquittal, acquittals, acquittance [ ... and 883 more ]  
#> - [CONSTRAINING]:  
#> - abide, abiding, bound, bounded, commit, commitment, commitments, commits, committed, committing, compel, compelled, compelling, compels, comply, compulsion, compulsory, confine, confined, confinement [ ... and 164 more ]  
#> - [SUPERFLUOUS]:  
#> - aegis, amorphous, anticipatory, appertaining, assimilate, assimilating, assimilation, bifurcated, bifurcation, cessions, cognizable, concomitant, correlative, deconsolidation, delineation, demonstrable, demonstrably, derecognized, derecognizes, derivatively [ ... and 36 more ]  
#> [ reached max\_nkey ... 3 more keys ]

A szentiment szótár 9 kategóriából áll. A legtöbb kulcsszó a negatív dimenzióhoz van (2355).

A munkamenet hasonló a Magyar Nemzetes példához:

1. adat betöltés
2. szövegtisztítás
3. korpusz
4. tokenek
5. kulcs kontextuális tokenek szűrése
6. dfm előállítás és szentiment számítás
7. az eredmény vizualizálása, további felhasználása

mnb\_pr <- read\_csv("data/mnb\_pr\_corpus.csv")  
  
summary(mnb\_pr)  
#> date text id year   
#> Min. :2005-01-24 Length:180 Min. : 1.00 Min. :2005   
#> 1st Qu.:2008-10-14 Class :character 1st Qu.: 45.75 1st Qu.:2008   
#> Median :2012-07-10 Mode :character Median : 90.50 Median :2012   
#> Mean :2012-07-08 Mean : 90.50 Mean :2012   
#> 3rd Qu.:2016-03-30 3rd Qu.:135.25 3rd Qu.:2016   
#> Max. :2019-12-17 Max. :180.00 Max. :2019

Az adatbázisunk 180 megfigyelésből és 4 változóból áll. Az egyetlen lényeges dokumentum meta adat itt is a szövegek megjelenési ideje.

A szövegeket ugyanazokkal a standard eszközökkel kezeljük mint a Magyar Nemzet esetében. Érdemes minden esetben ellenőrizni, hogy az R kód amit használunk az tényleg azt csinálja-e mint amit szeretnénk hogy csináljon. Ez hatványozottan igaz abban az esetben, amikor szövegekkel és regular expressionökkel dolgozunk.

mnb\_tiszta <- mnb\_pr %>%  
 mutate(  
 text = str\_remove\_all(string = text, pattern = "[:cntrl:]"),  
 text = str\_remove\_all(string = text, pattern = "[:punct:]"),  
 text = str\_remove\_all(string = text, pattern = "[:digit:]"),  
 text = str\_to\_lower(text),  
 text = str\_trim(text),  
 text = str\_squish(text)  
 )

Miután rendelkezésre állnak a tiszta dokumentumaink, egy karaktervektorba gyűjtjük azokat a kulcsszavakat amelyek környékén szeretnénk megfigyelni a szentiment alakulását. A példa kedvéért mi az unemp\*, growth, gdp, inflation\* szótöveket és szavakat választottuk. A tokens\_keep() megtartja a kulcsszavainkat és egy általunk megadott +/- n tokenes környezetüket (jelen esetben 10). A szentiment elemzést pedig már ezen a jóval kisebb mátrixon fogjuk lefuttatni. A phrase() segítségével több szóból álló kifejezéséket is vizsgálhatunk. Ilyen szókapcsolat például az *Európai Unió* is, ahol lényeges hogy egyben kezeljük a két szót.

mnb\_corpus <- corpus(mnb\_tiszta)  
  
gazdasag <- c("unemp\*", "growth", "gdp", "inflation\*", "inflation expectation\*")  
  
mnb\_token <- tokens(mnb\_corpus) %>%  
 tokens\_keep(pattern = phrase(gazdasag), window = 10)

A szentimentet most is egy súlyozott dfm-ből számoljuk. A kész eredményt hozzáadjuk a korpuszhoz majd data framet hozunk létre belőle. A 9 kategóriából 5-öt adunk választunk csak ki, amelyeknek jegybanki környezetben értelmezhető tartalma van.

mnb\_szentiment <- tokens\_lookup(mnb\_token, dictionary = penzugy\_szentiment) %>%  
 dfm() %>%  
 dfm\_tfidf()  
  
docvars(mnb\_corpus, "negative") <- as.numeric(mnb\_szentiment[, "negative"])  
docvars(mnb\_corpus, "positive") <- as.numeric(mnb\_szentiment[, "positive"])  
docvars(mnb\_corpus, "uncertainty") <- as.numeric(mnb\_szentiment[, "uncertainty"])  
docvars(mnb\_corpus, "constraining") <- as.numeric(mnb\_szentiment[, "constraining"])  
docvars(mnb\_corpus, "superfluous") <- as.numeric(mnb\_szentiment[, "superfluous"])  
  
mnb\_df <- convert(mnb\_corpus, to = "data.frame")

A célunk hogy szentiment kategóriánkénti bontásban mutassuk be az elemzésünk eredményét, de előtte egy kicsit alakítani kell a data frame-n, hogy a második fejezetben is tárgyalt *tidy* formára hozzuk. A különböző szentiment értékeket tartalmazó oszlopokat fogjuk átrendezni úgy hogy kreálunk egy “sent\_type” változót ahol a kategória nevet fogjuk eltárolni és egy “sent\_score” változót, ahol a szentiment értéket. Ehhez a tidyr-ben található pivot\_longer() -t használjuk.

mnb\_df <- mnb\_df %>%  
 pivot\_longer(  
 cols = negative:superfluous,  
 names\_to = "sent\_type",  
 values\_to = "sent\_score"  
 )

Az átalakítás után már könnyedén tudjuk kategóriákra bontva megjeleníteni az MNB közlemények különböző látens dimenzióit. Fontos emlékezni arra, hogy ez az eredmény a kulcsszavaink +/- 10 tokenes környezetében lévő szavak szentimentjét mérik. Ami érdekes eredmény, hogy a felesleges “töltelék” szövegek (superflous kategória) szinte soha nem fordulnak elő a kulcsszavaink körül. A többi érték is nagyjából megfelel a várakozásainknak, habár a 2008-as gazdasági válság nem tűnik kiugró pontnak. Azonban a 2010 utáni európai válság már láthatóan megjelenik az idősorainkban.

A szótár amit használtunk az alapvetően az Egyesült Államokban a tőzsdén kereskedett cégek publikus beszámolóiból készült így elképzelhető, hogy egyes jegybanki környezetben sokat használt kifejezés nincs benne. A validálásra a kapott eredményeknek ezért is nagyon fontos, illetve érdemes azzal is tisztában lenni hogy a szótáras módszer nem tökéletes (ahogy az emberi vagy más gépi kódolás sem).

ggplot(mnb\_df, aes(date, sent\_score)) +  
 geom\_line() +  
 labs(  
 y = "Szentiment",  
 x = NULL  
 ) +  
 facet\_wrap(~sent\_type, ncol = 2)

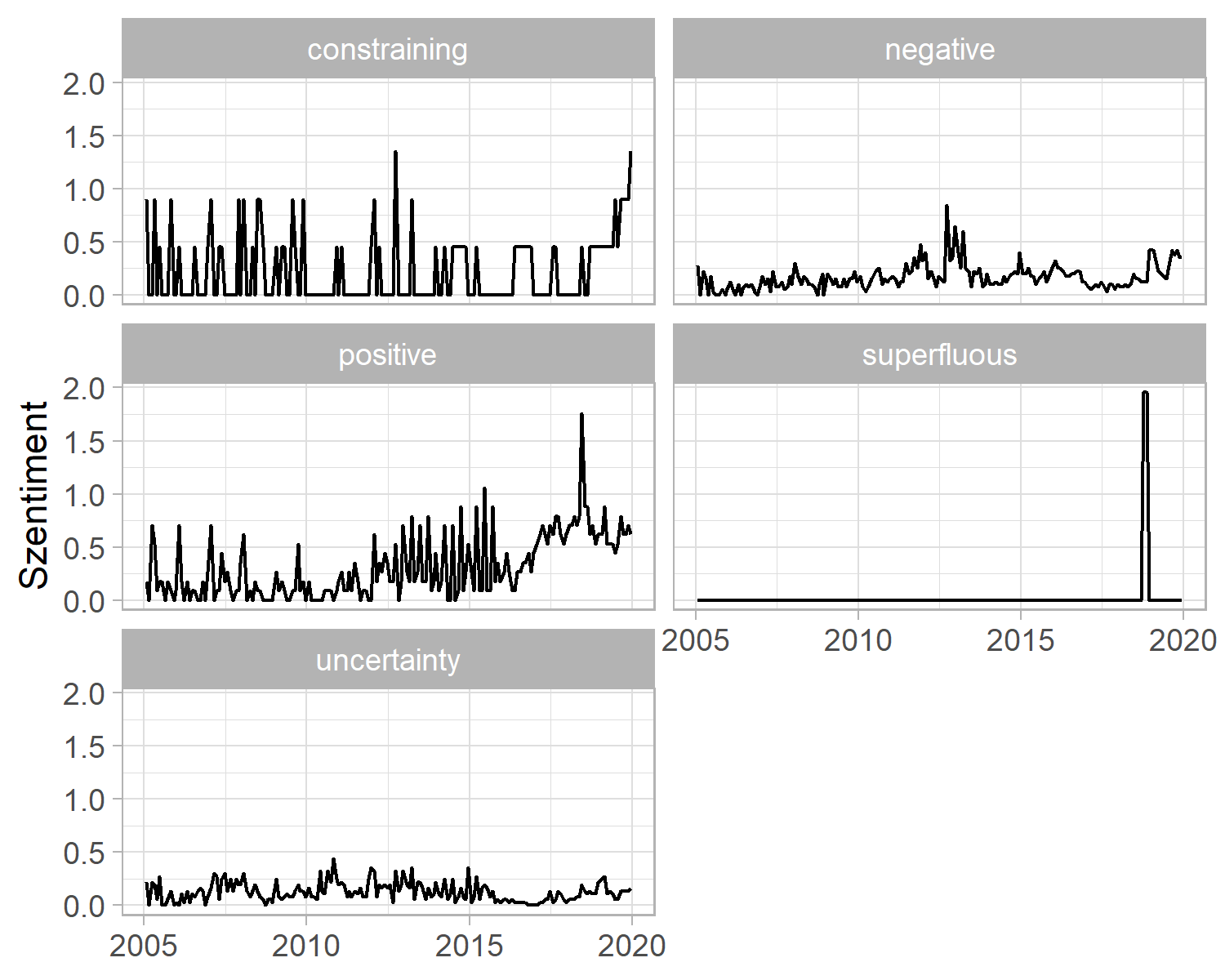


Figure 6.2: Magyar Nemzeti Bank közleményeinek szentimentje

# 7 Felügyelet nélküli tanulás: Topik modellezés magyar törvényszövegeken

A klaszterezés egy adathalmaz pontjainak, rekordjainak hasonlóság alapján való csoportosítása, ami szinte minden nagyméretű adathalmaz leíró modellezésére alkalmas. A klaszterezés során az adatpontokat diszjunkt halmazokba, azaz klaszterekbe soroljuk, hogy az elemeknek egy olyan partíciója jöjjön létre, amelyben a közös csoportokba kerülő elempárok lényegesen hasonlóbbak egymáshoz, mint azok a pontpárok, melyek két különböző csoportba sorolódtak. Klaszterezés során a megfelelő csoportok kialakítása nem egyértelmű feladat, mivel a különböző adatok eltérő jelentése és felhasználása miatt adathalmazonként más szempontokat kell figyelembe vennünk. Egy klaszterezési feladat megoldásához ismernünk kell a különböző algoritmusok alapvető tulajdonságait és mindig szükség van az eredményként kapott klaszterezés kiértékelésére. Mivel egy klaszterezés az adatpontok hasonlóságából indul ki, ezért az eljárás során az első fontos lépés az adatpontok páronkénti hasonlóságát lehető legjobban megragadó hasonlósági függvény kiválasztása ([Tan, Steinbach, and Kumar 2011](#ref-tanBevezetesAzAdatbanyaszatba2011)). Számos klaszterezési eljárás létezik, melyek között az egyik leggyakoribb különbségtétel, hogy a klaszterek egymásba ágyazottak vagy sem. Ez alapján beszélhetünk hierarchikus és felosztó klaszterezésről.

A hierarchikus klaszterezés egymásba ágyazott klaszterek egy fába rendezett halmaza, azaz ahol a klaszterek alklaszterekkel rendelkeznek. A fa minden csúcsa (klasztere), a levélcsúcsokat kivéve, a gyermekei (alklaszterei) uniója, és a fa gyökere az összes objektumot tartalmazó klaszter. Felosztó (partitional) klaszterezés esetén az adathalmazt olyan, nem átfedő alcsoportokra bontjuk, ahol minden adatobjektum pontosan egy részhalmazba kerül ([Tan, Steinbach, and Kumar 2011](#ref-tanBevezetesAzAdatbanyaszatba2011); [Tikk 2007](#ref-tikkSzovegbanyaszat2007)). A klaszterezési eljárások között aszerint is különbséget tehetünk, hogy azok egy objektumot csak egy vagy több klaszterbe is beilleszthetnek. Ez alapján beszélhetünk kizáró (exclusive), illetve nem-kizáró (non exclusive), vagy átfedő (overlapping) klaszterezésről. Az előbbi minden objektumot csak egyetlen klaszterhez rendel hozzá, az utóbbi esetén egy pont több klaszterbe is beleillik. Fuzzy klaszterezés esetén minden objektum minden klaszterbe beletartozik egy tagsági súly erejéig, melynek értéke 0 (egyáltalán nem tartozik bele) és 1 (teljesen beletartozik) közé esik. A klasztereknek is különböző típusai vannak, így beszélhetünk prototípus-alapú, gráf-alapú vagy sűrűség-alapú klaszterekről.

A prototípus-alapú klaszter olyan objektumokat tartalmazó halmaz, amelynek mindegyik objektuma jobban hasonlít a klasztert definiáló objektumhoz, mint bármelyik másik klasztert definiáló objektumhoz. A prototípus-alapú klaszter klaszterek közül a K-közép klaszter az egyik leggyakrabban alkalmazott. A K-közép klaszterezési módszer első lépése K darab kezdő középpontot kijelölése, ahol K a klaszterek kívánt számával egyenlő. Ezután minden adatpontot a hozzá legközelebb eső középponthoz rendelünk. Az így képzett csoportok lesznek a kiinduló klaszterek. Ezután újra meghatározzuk mindegyik klaszter középpontját a klaszterhez rendelt pontok alapján. A hozzárendelési és frissítési lépéseket felváltva folytatjuk addig, amíg egyetlen pont sem vált klasztert, vagy ameddig a középpontok ugyanazok nem maradnak ([Tan, Steinbach, and Kumar 2011](#ref-tanBevezetesAzAdatbanyaszatba2011)).

## 7.1 K közép klaszterezés kvalitatív adatokkal

A K közép klaszterezés tehát a dokumentumokat alkotó szavak alapján keresi meg a felhasználó által megadott számú (K) klasztert, amelyeket a középpontjaik képviselnek, és így rendezi a dokumentumokat csoportokba. A klaszterezés vagy csoportosítás egy induktív kategorizálás, ami akkor hasznos, amikor nem állnak a kutató rendelkezésére előzetesen ismert csoportok, amelyek szerint a vizsgált dokumentumokat rendezni tudná. Hiszen ebben az esetben a korpusz elemeinek rendezéséhez nem határozunk meg előzetesen csoportokat, hanem az eljárás során olyan különálló csoportokat hozunk létre a dokumentumokból, amelynek tagjai valamilyen szempontból hasonlítanak egymásra. A csoportosítás legfőbb célja az, hogy az egy csoportba kerülő szövegek minél inkább hasonlítsanak egymásra, miközben a különböző csoportba kerülők minél inkább eltérjenek egymástól. Azaz klaszterezésnél nem egy-egy szöveg jellemzőire vagyunk kíváncsiak, hanem arra, hogy a szövegek egy-egy csoportja milyen hasonlóságokkal bír ([Tikk 2007](#ref-tikkSzovegbanyaszat2007); [Burtejin 2016](#Xd2fcd86b9e924ef79c0251edbf4003b7c0326b4)). A gépi kódolással végzett klaszterezés egy felügyelet nélküli tanulás, mely a szöveg tulajdonságaiból tanul, anélkül, hogy előre meghatározott csoportokat ismerne. Alkalmazása során a dokumentum tulajdonságait és a modell becsléseit felhasználva jönnek létre a különböző kategóriák, melyekhez később hozzárendeli a szöveget ([Grimmer and Stewart 2013](#ref-grimmer2013text)). Az osztályozással ellentétben a csoportosítás esetén tehát nincs ismert „címkékkel" ellátott kategóriarendszer vagy olyan minta, mint az osztályozás esetében a tanítókörnyezet, amiből tanulva a modellt fel lehet építeni ([Tikk 2007](#ref-tikkSzovegbanyaszat2007)). A gépi kódolással végzett csoportosítás (klaszterezés) esetén a kutató feladata a megfelelő csoportosító mechanizmus kiválasztása, mely alapján egy program végzi el a szövegek különböző kategóriákba sorolását. Ezt követi a hasonló szövegeket tömörítő csoportok elnevezésének lépése. A több dokumentumból álló korpuszok esetében a gépi klaszterelemzés különösen eredményes és költséghatékony lehet, mivel egy nagy korpusz vizsgálata sok erőforrást igényel ([Grimmer and Stewart 2013, 1](#ref-grimmer2013text) o.)

A klaszterezés bemutatásához a rendszerváltás utáni magyar miniszterelnökök egy-egy véletlenszerűen kiválasztott beszédét használjuk.

library(readr)  
library(dplyr)  
library(purrr)  
library(stringr)  
library(readtext)  
library(quanteda)  
library(tidytext)  
library(ggplot2)  
library(topicmodels)  
library(factoextra)  
library(stm)  
library(igraph)

A beszédek szövege meglehetősen tiszta, ezért az egyszerűség kedvéért a most kihagyjuk a szövegtisztítás lépéseit. Az elemzés első lépéseként a .csv fájlból beolvasott szövegeinkből a quanteda csomaggal korpuszt hozunk létre, majd abból egy dokumentum-kifejezés mátrixot készítünk a dfm() függvénnyel. Láthatjuk, hogy márixunk 7 megfigyelést és 4 változót tartalmaz.

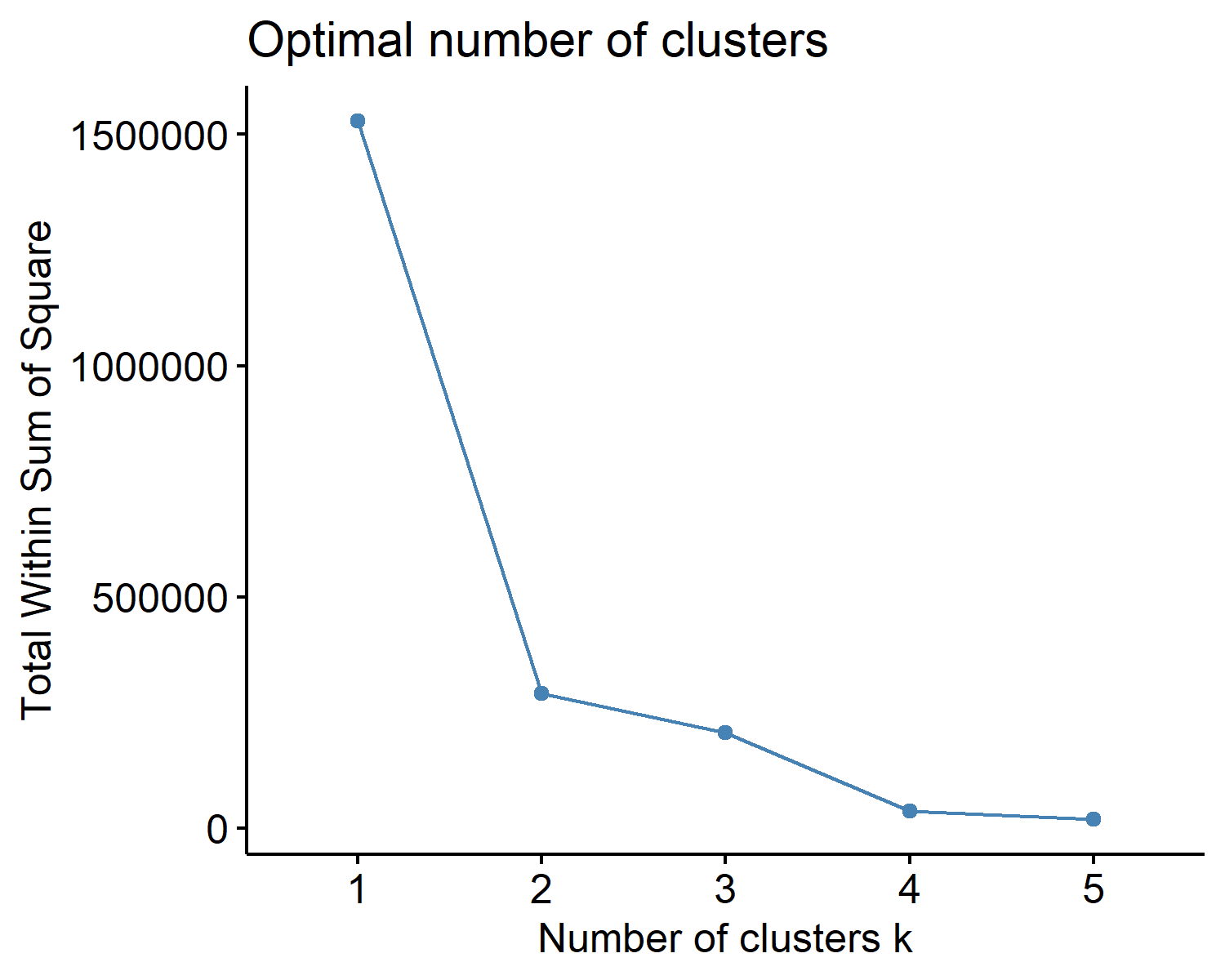
beszedek <- read\_csv("data/miniszterelnokok.csv")  
  
beszedek\_corpus <- corpus(beszedek)  
  
beszedek\_dfm <- dfm(beszedek\_corpus)

A beszédek klaszterekbe rendezését az R egyik alapfüggvénye végzi, a kmeans. Első lépésben 3 klasztert készítünk. A table() függvénnyel megnézhetjük hogy egy-egy csoportba hány dokumentum került.

beszedek\_klaszter <- kmeans(beszedek\_dfm, centers = 2)  
  
  
table(beszedek\_klaszter$cluster)  
#>   
#> 1 2   
#> 5 2

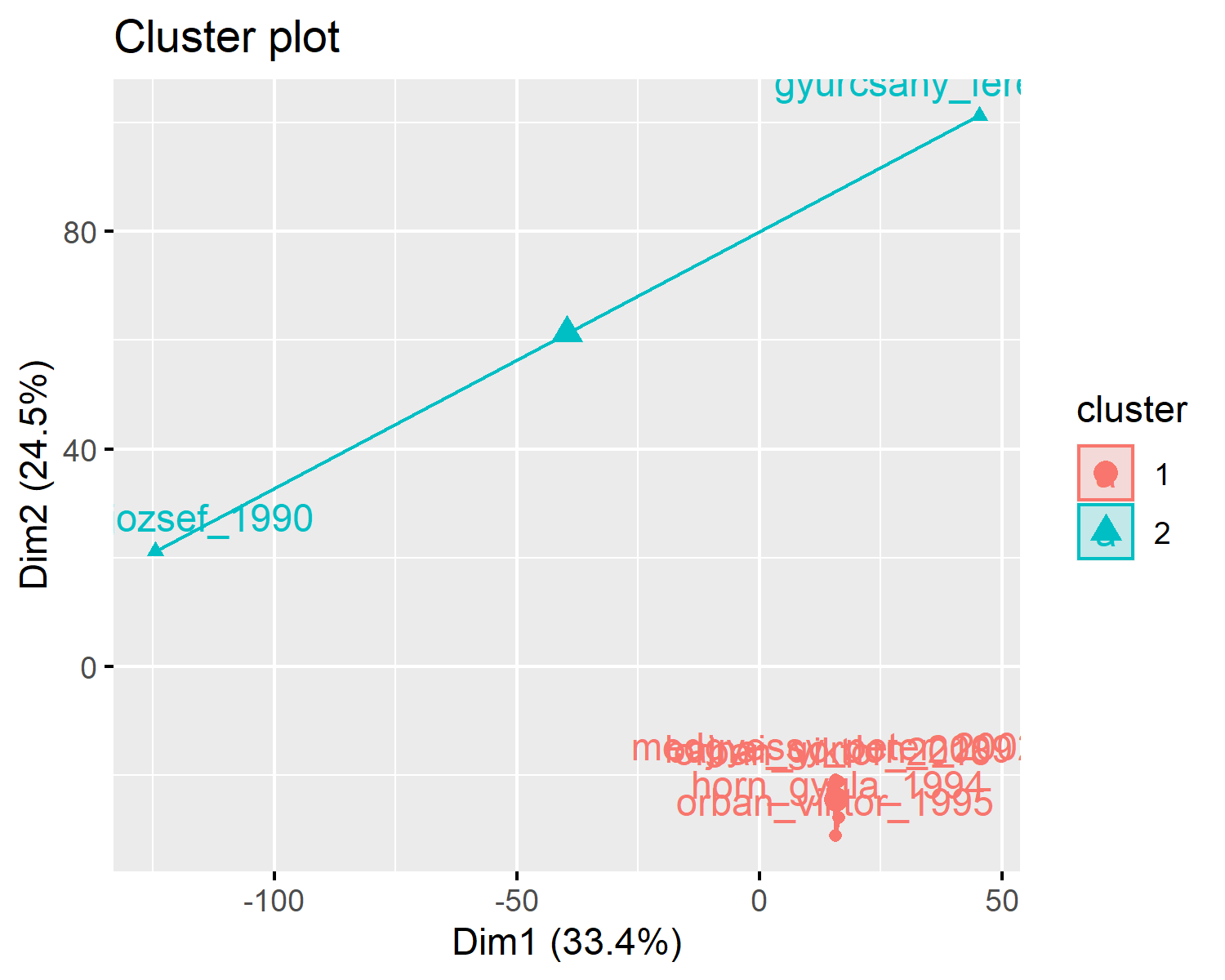
A felügyelet nélküli klasszifikáció nagy kérdése, hogy hány klasztert alakítsunk ki, hogy megközelítsük a valóságot, és ne csak mesterségesen kreáljunk csoportokat. Ez ugyanis azzal a kockázattal jár, hogy ténylegesen nem létező csoportok is létrejönnek. A klaszterek optimális számának meghatározására kvalitatív és kvantitatív lehetőségeink is vannak. A következőkben az utóbbira mutatunk példát, amihez a factoextra csomagot használjuk A lenti ábra azt mutatja hogy a klasztereken belüli négyzetösszegek hogyan változnak a **k** paraméter változásának függvényében. Minél kisebb a klasztereken belüli négyzetösszegek érteke, annál közelebbi pontok tartoznak össze, így a kisebb értékekéel jobban definialt klasztereket kapunk. Az ábra alapjén tehát az ideális **k** 4 vagy 2, attól fuggően, hogy milyen feltevésekkel élünk a kutatasunk során. A 2-es értek azért lehet jó, mert a értékek esetén a négyzetösszegek érteke nem csökken drasztikusan és a korpuszunk alapjan a a két (“jobb-bal”) klaszter kvalitativ alapon is jol definiálható. A pedig azért lehet jó, mert utánna gyakorlatilag nem változik a kapott négyzetösszeg, ami azt jelzi, hogy a tovabbi klaszterek hozzáadásaval nem lesz pontosabb a csoportosítás.

fviz\_nbclust(as.matrix(beszedek\_dfm), kmeans, method = "wss", k.max = 5)



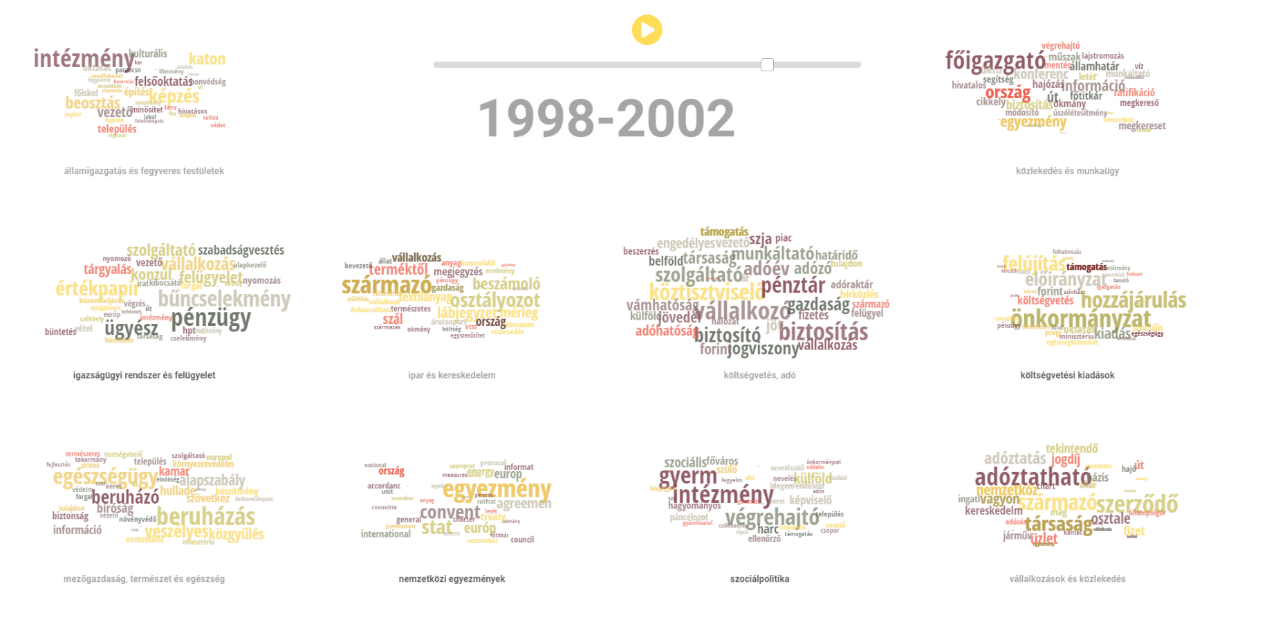
A kialakított csoportokat vizuálisan is megjeleníthetjük.

fviz\_cluster(beszedek\_klaszter, data = beszedek\_dfm)



## 7.2 Látens Dirichlet Allokáció topik modellek[[27]](#footnote-132)

A topik-modellezés a dokumentumok téma-klasztereinek meghatározására szolgáló valószínűség-alapú eljárás, amely szó-gyakoriságot állapít meg minden témához, és minden dokumentumhoz hozzárendeli az adott témák valószínűségét. A topik modellezés egy felügyelet nélküli tanulási módszer, amely során az alkalmazott algoritmus a dokumentum tulajdonságait és a modell becsléseit felhasználva hoz létre különböző kategóriákat, melyekhez később hozzárendeli a szöveget ([Tikk 2007](#ref-tikkSzovegbanyaszat2007); [Grimmer and Stewart 2013](#ref-grimmer2013text); [Burtejin 2016](#Xd2fcd86b9e924ef79c0251edbf4003b7c0326b4)). Az egyik leggyakrabban alkalmazott topik modellezési eljárás, a Látens Dirichlet Allokáció (LDA) alapja az a feltételezés, hogy minden korpusz topikok/témák keverékéből áll, ezen témák pedig statisztikailag a korpusz szókészlete valószínűségi függvényeinek (eloszlásának) tekinthetőek ([Blei, Ng, and Jordan 2003](#ref-bleiLatentDirichletAllocation2003)). Az LDA a korpusz dokumentumainak csoportosítása során az egyes dokumentumokhoz topik szavakat rendel, a topikok megbecsléséhez pedig a szavak együttes megjelenését vizsgálja a dokumentum egészében. Az LDA algoritmusnak előzetesen meg kell adni a keresett klaszterek (azaz a keresett topikok) számát, ezt követően a dokumentumhalmazban szereplő szavak eloszlása alapján az algoritmus azonosítja a kulcsszavakat, amelyek eloszlása kirajzolja a topikokat ([Blei, Ng, and Jordan 2003](#ref-bleiLatentDirichletAllocation2003); [Burtejin 2016](#Xd2fcd86b9e924ef79c0251edbf4003b7c0326b4); [Jacobi, Van Atteveldt, and Welbers 2016](#ref-jacobiQuantitativeAnalysisLarge2016)).



A következőkben a magyar törvények korpuszán szemléltetjük a topik modellezés módszerét, hogy a mesterséges intelligencia segítségével feltárjuk a korpuszon belüli rejtett összefüggéseket. A korábban leírtak szerint tehát nincsenek előre meghatározott kategóriáink, dokumentumainkat a klaszterezés segítségével szeretnénk csoportosítani. Egy-egy dokumentumban keveredhetnek a témák és az azokat reprezentáló szavak. Mivel ugyanaz a szó több topikhoz is kapcsolódhat, így az eljárás komplex elemzési lehetőséget nyújt, az egy szövegen belül témák és akár azok dokumentumon belüli súlyának azonosítására. Példánkban csak a korpusz egy részén szemléltetjük a topik modellezést, a teljes korpusz és annak elemzéséhez szükséges kód elérhető az alábbi GitHub linken: <https://github.com/poltextlab>

Az alábbiakban 1998-2002 és a 2002-2006-os parlamenti ciklus 1032 törvényszövegének topik modellezését és a szükséges előkészítő, korpusztisztító lépéseket mutatjuk be. A fájlokat töltsük be az R által használt munkakönyvtárba.[[28]](#footnote-135)

Töltsük be az elemezni kívánt csv fájlt, megadva az elérési útvonalát.

torvenyek <- read\_csv("data/lawtext\_1998\_2006.csv")

Az előző fejezetekben láthattuk hogy hogyan lehet használni a stringr csomagot a szövegtisztításra. A lépések a már megismert sztenderd folyamatot követik: számok, központozás, sortörések, extra szóközök eltávolítása, illetve a szöveg kisbetűsítése. Az eddigieket további szövegtisztító lépésekkel is kiegészíthetjük. Olyan elemek esetében, amelyek nem feltétlenül különálló szavak és el akarjuk távolítani őket a korpuszból szintén az str\_remove\_all() a legegyszerűbb megoldás.

torvenyek\_tiszta <- torvenyek %>%  
 mutate(  
 text = str\_remove\_all(string = text, pattern = "[:cntrl:]"),  
 text = str\_remove\_all(string = text, pattern = "[:punct:]"),  
 text = str\_remove\_all(string = text, pattern = "[:digit:]"),  
 text = str\_to\_lower(text),  
 text = str\_trim(text),  
 text = str\_squish(text),  
 text = str\_remove\_all(string = text, pattern = "’"),  
 text = str\_remove\_all(string = text, pattern = "…"),  
 text = str\_remove\_all(string = text, pattern = "–"),  
 text = str\_remove\_all(string = text, pattern = "“"),  
 text = str\_remove\_all(string = text, pattern = "”"),  
 text = str\_remove\_all(string = text, pattern = "„"),  
 text = str\_remove\_all(string = text, pattern = "«"),  
 text = str\_remove\_all(string = text, pattern = "»"),  
 text = str\_remove\_all(string = text, pattern = "§"),  
 text = str\_remove\_all(string = text, pattern = "°"),  
 text = str\_remove\_all(string = text, pattern = "<U+25A1>"),  
 text = str\_remove\_all(string = text, pattern = "@")  
 )

A dokumentum változókat egy külön fájlból adjuk hozzá, ami a törvények keletkezési évét tartalmazza, illetve hogy melyik kormányzati ciklusban születtek. Mindkét adatbázisban egy közös egyedi azonosító jelöli az egyes törvényeket, így ki tudjuk használni a dplyr left\_join() függvényét, ami hatékonyan és gyorsan kapcsol össze adatbázisokat közös egyedi azonosító mentén. Jelen esetben ez az egyedi azonosító a txt\_filename oszlopból fog elkészülni, amely a törvények neveit tartalmazza. Első lépésben betöltjük a meta adatokat tartalmazó .csv fájlt, majd a .txt rész előtti törvényneveket tartjuk csak meg a létrehozott doc\_id- oszlopban. A [^\\.]\* regular expression itt a string elejétől indulva kijelöl mindent az elso . karakterig. Az str\_extract() pedig ezt a kijelölt string szakaszt (ami a törvények neve) menti át az új változónkba.

torveny\_meta <- read\_csv("data/cap\_law\_meta.csv")  
  
torveny\_meta <- torveny\_meta %>%  
 mutate(doc\_id = str\_extract(txt\_filename, "[^\\.]\*")) %>%  
 select(-txt\_filename)  
  
head(torveny\_meta, 5)  
#> # A tibble: 5 x 4  
#> year electoral\_cycle majortopic doc\_id   
#> <dbl> <chr> <dbl> <chr>   
#> 1 1998 1998-2002 13 1998XXXV   
#> 2 1998 1998-2002 20 1998XXXVI   
#> 3 1998 1998-2002 3 1998XXXVII   
#> 4 1998 1998-2002 6 1998XXXVIII  
#> 5 1998 1998-2002 13 1998XXXIX

Végül összefűzzük a dokumentumokat és a meta adatokat tartalmazó data frameket.

torveny\_final <- left\_join(torvenyek\_tiszta, torveny\_meta, by = "doc\_id")

Majd létrehozzuk a korpuszt és ellenőrizzük azt.

#> Text Types Tokens Sentences year electoral\_cycle majortopic  
#> 1 1998L 2879 9628 1 1998 1998-2002 3  
#> 2 1998LI 352 680 1 1998 1998-2002 20  
#> 3 1998LII 446 992 1 1998 1998-2002 9  
#> 4 1998LIII 126 221 1 1998 1998-2002 9  
#> 5 1998LIV 835 2013 1 1998 1998-2002 9

Az RStudio environments fülén láthatjuk, hogy egy 1032 elemből álló korpusz jött létre, amelynek tartalmát a summary() paranccsal kiíratva a console ablakban megjelenik a dokumentumok listája és a főbb leíró statisztikai adatok (egyedi szavak - types; szószám - tokens; mondatok - sentences). Az előbbi fejezettől eltérően most a tokenizálás során is végzünk még egy kis tisztítást: a felesleges stop szavakat kitöröljük a tokens\_remove() és stopwords() kombinálásával. A quanteda tartalmaz egy beépített magyar stopszó szótárat. A második lépésben szótövesítjük a tokeneket a tokens\_words() használatával, ami szintén képes a magyar nyelvű szövegeket kezelni.

Szükség esetén a beépített magyar nyelvű stopszó szótárat saját stopszavakkal is kiegészíthetjük. Ehhez először csv fájlba el kell mentenünk a stopszavakat, majd a csv fájlt be kell olvasnunk. Az pull() egy karaktervektort fog kreálni a data frame text oszlopából.

custom\_stopwords <- readtext("data/custom\_legal\_stopwords.csv", encoding = "UTF8") %>%  
 pull(text)

Mivel a korpusz ellenőrzése során találunk még olyan kifejezéseket, amelyeket el szeretnénk távolítani, ezeket is kiszűrjük.

custom\_stopwords\_egyeb <- c("lábjegyzet", "országgyűlés", "ülésnap")

Aztán pedig a korábban már megismert **pipe operáror** használatával elkészítjük a token objektumunkat. A szótövesített tokeneket egy külön objektumban tároljuk, mert gyakran előfordul hogy később vissza kell térnünk az eredeti token objektumhoz, hogy egyéb műveleteket végezzünk el, például további stopszavakat távolítsunk el.

torvenyek\_tokens <- tokens(torvenyek\_corpus) %>%  
 tokens\_remove(stopwords("hungarian")) %>%  
 tokens\_remove(custom\_stopwords) %>%  
 tokens\_remove(custom\_stopwords\_egyeb) %>%  
 tokens\_wordstem(language = "hun")

Végül eltávolítjuk a dokumentum kifejezés mátrixból a túl gyakori kifejezéseket. A dfm\_trim() függvénnyel a nagyon ritka és nagyon gyakori szavak megjelenését kontrollálhatjuk. A termfreq\_type opció "prop" akkor 0 és 1.0 közötti értéket vehetnek fel a max\_termfreq/docfreq és min\_termfreq/docfreq paraméterek. A lenti példában azokat a tokeneket tartjuk meg, amelyek legalább egyszer előfordulnak ezer dokumentumonként (így kizárva a nagyon ritka kifejezéseket).

torvenyek\_dfm <- dfm(torvenyek\_tokens) %>%  
 dfm\_trim(min\_termfreq = 0.001, termfreq\_type = "prop")

A szövegtisztító lépesek eredményét úgy ellenőrizhetjük, hogy az 2. fejezetben bemutatottak szerint szógyakorisági listát készítünk a korpuszban maradt kifejezésekről. Itt kihasználhatjuk a korpuszunkban lévő meta adatokat és megnézhetjük ciklus szerinti bontásban a szófrekvencia ábrát. Az ábránál figyeljünk arra hogy a tidytext reorder\_within függvényét használjuk, ami egy nagyon hasznos megoldás a csoportosított sorrendbe rendezésre a ggplot2 ábránál.

top\_tokens <- textstat\_frequency(torvenyek\_dfm, n = 15, groups = docvars(torvenyek\_dfm,   
 field = "electoral\_cycle"))  
  
ggplot(top\_tokens, aes(reorder\_within(feature, frequency, group), frequency)) + geom\_point(aes(shape = group),   
 size = 2) + coord\_flip() + labs(y = NULL, x = "szófrekvencia") + facet\_wrap(~group,   
 nrow = 2, scales = "free") + tidytext::scale\_x\_reordered()

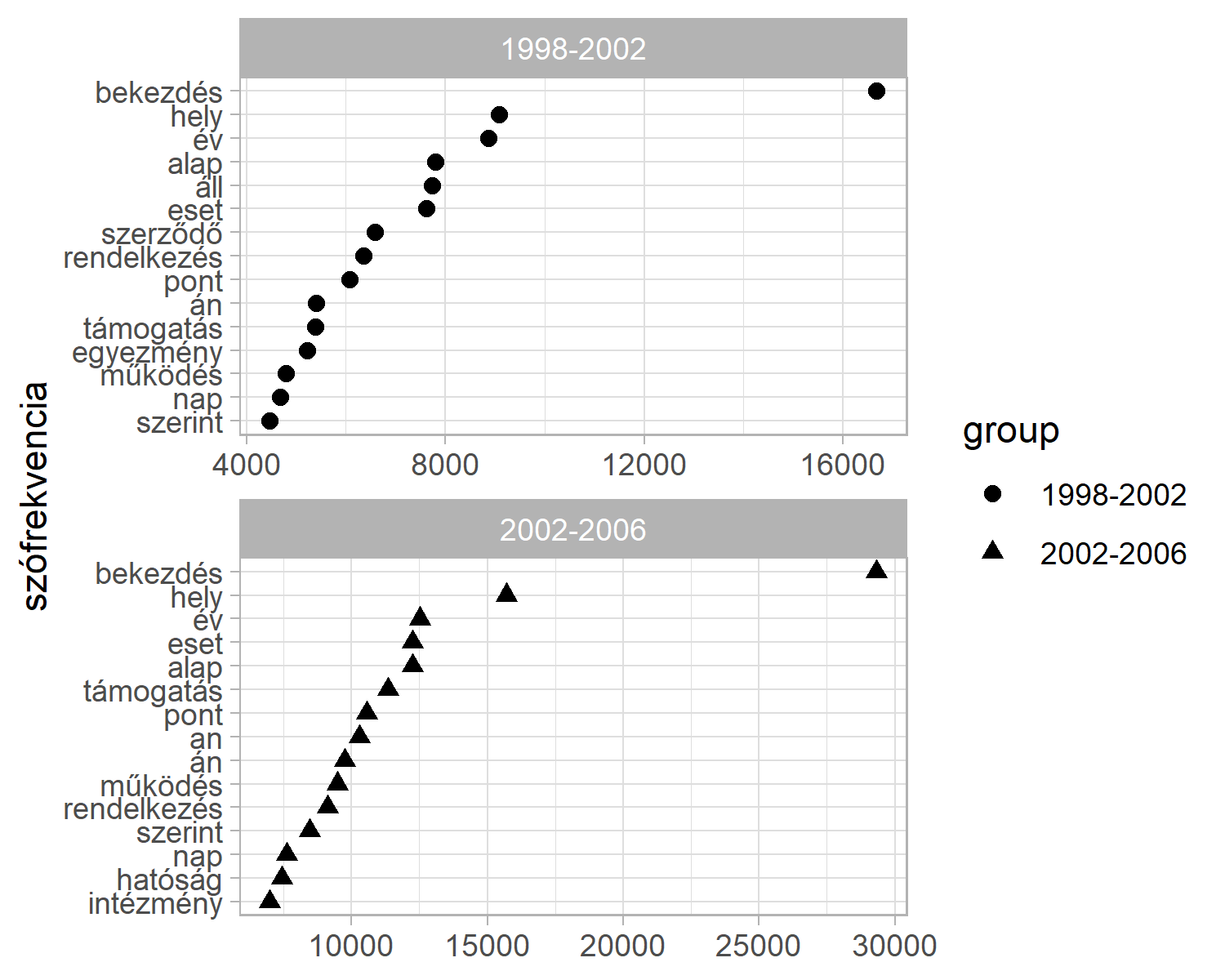


Figure 7.1: A 15 leggyakoribb token a korpuszban

A szövegtisztító lépéseket később újabbakkal is kiegészíthetjük, ha észrevesszük, hogy az elemzést zavaró tisztítási lépés maradt ki. Ilyen esetben tovább tisztíthatjuk a korpuszt, majd újra lefuttathatjuk az elemzést. Például, ha szükséges, további stopszavak eltávolítását is elvégezhetjük egy újabb stopszólista hozzáadásával. Ilyenkor ugyanúgy járunk el, mint az előző stopszólista esetén, vagyis beolvassuk a munkakönyvtárban elhelyezett a csv fájlt, a beolvasott stopszólistából karakter vektort majd objektumot hozunk létre, végezetül pedig ezeket a szavakat is eltávolítjuk a kopuszból.

custom\_stopwords2 <- readtext("data/custom\_stopwords2.csv", encoding = "UTF8") %>%  
 pull(text)  
  
torvenyek\_tokens\_final <- torvenyek\_tokens %>%  
 tokens\_remove(custom\_stopwords2)

Ezután újra ellenőrizzük az eredményt.

torvenyek\_dfm\_final <- dfm(torvenyek\_tokens\_final) %>% dfm\_trim(min\_termfreq = 0.001,   
 termfreq\_type = "prop")  
  
top\_tokens\_final <- textstat\_frequency(torvenyek\_dfm\_final, n = 15, groups = docvars(torvenyek\_dfm,   
 field = "electoral\_cycle"))

És ezt egy ábrán is megjelenítjük.

ggplot(top\_tokens\_final, aes(reorder\_within(feature, frequency, group), frequency)) +  
 geom\_point(aes(shape = group), size = 2) +  
 coord\_flip() +  
 labs(  
 y = NULL,  
 x = "szófrekvencia"  
 ) +  
 facet\_wrap(~group, nrow = 2, scales = "free") +  
 tidytext::scale\_x\_reordered()

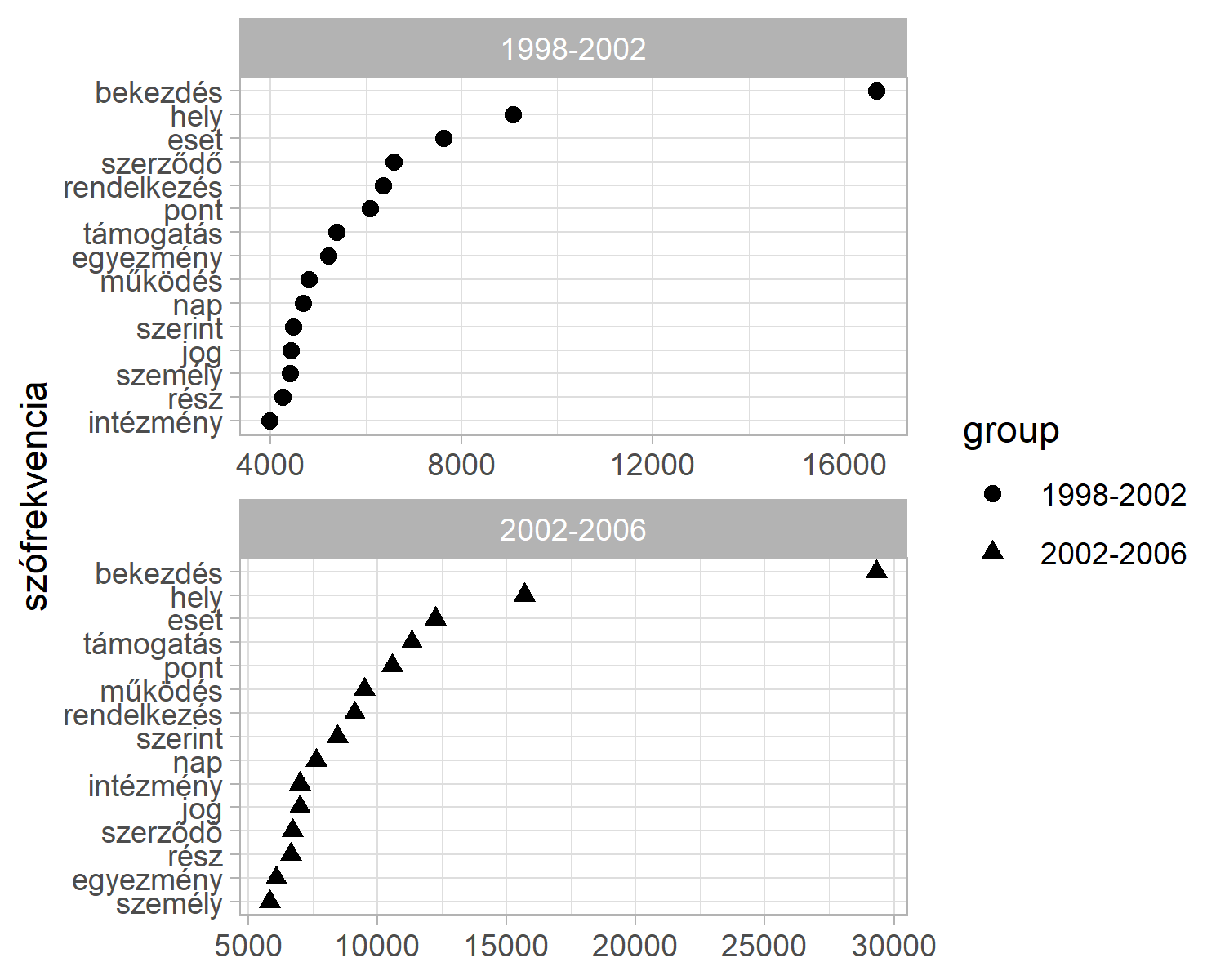


Figure 7.2: A 15 leggyakoribb token a korpuszban, a bovített stop szó listával

A szövegtisztító és korpusz előkészítő műveletek után következhet az LDA illesztése. Az alábbiakban az LDA illesztés két módszerét a VEM-et és a Gibbs-et mutatjuk be. A modell minkét módszer esetén ugyanaz, a különbség a következtetés módjában van. A VEM módszer variációs következtetés, míg a Gibbs mintavételen alapuló következtetés. ([Blei, Ng, and Jordan 2003](#ref-bleiLatentDirichletAllocation2003); [Griffiths and Steyvers 2004](#ref-griffithsFindingScientificTopics2004); [Phan, Nguyen, and Horiguchi 2008](#ref-phanLearningClassifyShort2008))

A két modell illesztése nagyon hasonló, meg kell adnunk, az elemezni kívánt dfm nevét, majd a *„k"* értékét, ami egyenlő az általunk létrehozni kívánt topikok számával, ezt követően meg kell jelölnünk, hogy a VEM vagy a Gibbs módszert alkalmazzuk. A set.seed() funkció az R véletlen szám generátor magjának beállítására szolgál, ami ahhoz kell, hogy a kapott eredmények, ábrák, stb. pontosan reprodukálhatóak legyenek. A set.seed() bármilyen tetszőleges egész szám lehet. Mivel az elemzésünk célja a két ciklus jogalkotásának összehasonlítása, a korpuszunkat két alkorpuszra bontjuk,ehhez a dokumentumok kormányzati ciklus azonosítóját használjuk fel. A dokumentum változók alapján a dfm\_subset()parancs segítségével választjuk szét a már elkészült és tisztított mátrixunkat.

dfm\_98\_02 <- dfm\_subset(torvenyek\_dfm\_final, electoral\_cycle == "1998-2002")  
  
dfm\_02\_06 <- dfm\_subset(torvenyek\_dfm\_final, electoral\_cycle == "2002-2006")

### 7.2.1 A „VEM" módszer alkalmazása a magyar törvények korpuszán

Saját korpuszunkon először a VEM a módszert alkalmazzuk, ahol k = 10 azaz a modell 10 témacsoportot alakít ki. Mint arról korábban már volt szó a *„k"* értékét szabadon változtathatjuk, aszerint hogy hány topik kialakítását szeretnénk. Ahogyan korábban arról már volt szó a *„k"* értékének meghatározása kutatói döntésen alapul, a modell futtatása során bevett gyakorlat a különböző *„k"* értékekkel való kísérletezés. Az elkészült modell kiértékelésr az elemzés elkészülte után a perplexity() függvény segítségével van lehetőségünk – ahol a *theta* az adott topikhoz való tartozás valószínűsége. A függvény a topikok által reprezentált elméleti szóeloszlásokat hasonlítja össze a szavak tényleges eloszlásával a dokumentumokban. A függvény értéke nem önmagában értelmezendő, hanem két modell összehasonlításában, ahol a legalacsonyabb *perplexity* (zavarodottság) értékkel rendelkező modellt tekintik a legjobbnak.[[29]](#footnote-139) Az illusztráció kedvéért lefuttatunk 4 LDA modellt az 1998-2002 kormányzati ciklushoz tartozó dfm-en. Az iterációhoz a purrr csomag map függvényét használtuk. Fontos megjegyezni, hogy minél nagyobb a korpuszunk annál több számítási kapacitásra van szükség (és annál tovább tart a számítás).

k\_topics <- c(5, 10, 15, 20)  
  
lda\_98\_02 <- k\_topics %>%  
 map(LDA, x = dfm\_98\_02, control = list(seed = 1234))  
  
  
perp\_df <- tibble(  
 k = k\_topics,  
 perplexity = map\_dbl(lda\_98\_02, perplexity)  
)

ggplot(perp\_df, aes(k, perplexity)) +  
 geom\_point() +  
 geom\_line() +  
 labs(  
 x = "k",  
 y = "Perplexity"  
 )

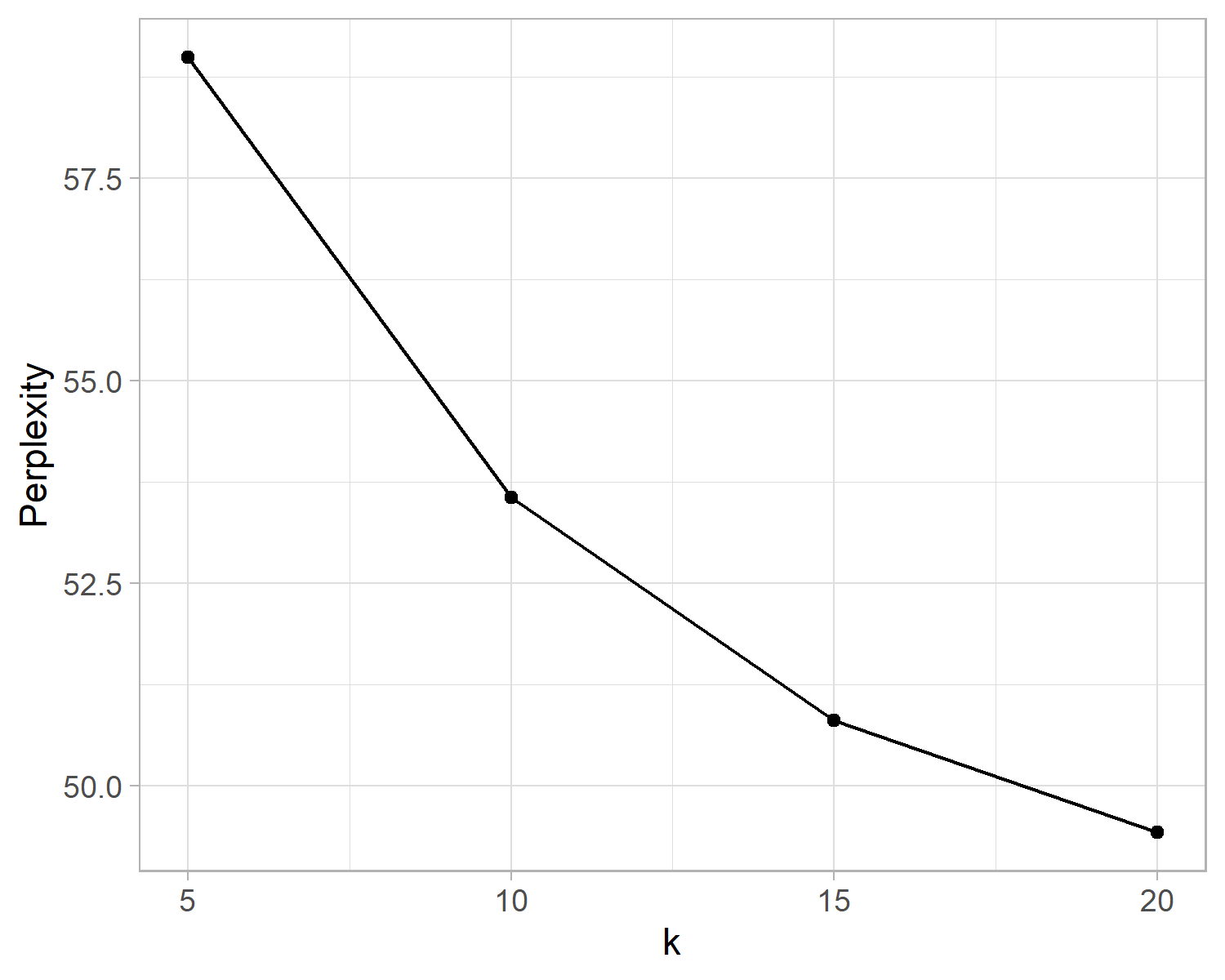


Figure 7.3: Perplexity változása a k függvényében

A perplexity pontszám alapján a 20 topikos modell szerepel a legjobban, de a megfelelő *k* kiválasztása a kutató kvalitatív döntésén múlik. A perplexity pontszám ehhez ad kvantitatív szempontokat.[[30]](#footnote-142)

A reprodukálhatóság és futási sebesség érdekében a fejezet további részeiben a *k* paraméternek 10-es értéket adunk. Ezzel lefuttatunk egy-egy modellt a két ciklusra.

vem\_98\_02 <- LDA(dfm\_98\_02, k = 10, method = "VEM", control = list(seed = 1234))  
  
vem\_02\_06 <- LDA(dfm\_02\_06, k = 10, method = "VEM", control = list(seed = 1234))

Ezt követően a modell által létrehozott topic-okat tidy formátumba tesszük és egyesítjük egy data frameben.[[31]](#footnote-143)

topics\_98\_02 <- tidy(vem\_98\_02, matrix = "beta") %>%  
 mutate(electoral\_cycle = "1998-2002")  
  
topics\_02\_06 <- tidy(vem\_02\_06, matrix = "beta") %>%  
 mutate(electoral\_cycle = "2002-2006")  
  
lda\_vem <- bind\_rows(topics\_98\_02, topics\_02\_06)

Majd listázzuk az egyes topikokhoz tartozó leggyakoribb kifejezéseket.

top\_terms <- lda\_vem %>%  
 group\_by(electoral\_cycle, topic) %>%  
 top\_n(5, beta) %>%  
 top\_n(5, term) %>%  
 ungroup() %>%  
 arrange(topic, -beta)

Majd a ggplot2 csomag segítségével ábrán is megjeleníthetjük az egyes topikok 10 legfontosabb kifejezését.

top\_terms %>%  
 filter(electoral\_cycle == "1998-2002") %>%  
 ggplot(aes(reorder\_within(term, beta, topic), beta)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~topic, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(  
 x = NULL,  
 y = expression(beta)  
 ) +  
 tidytext::scale\_x\_reordered()

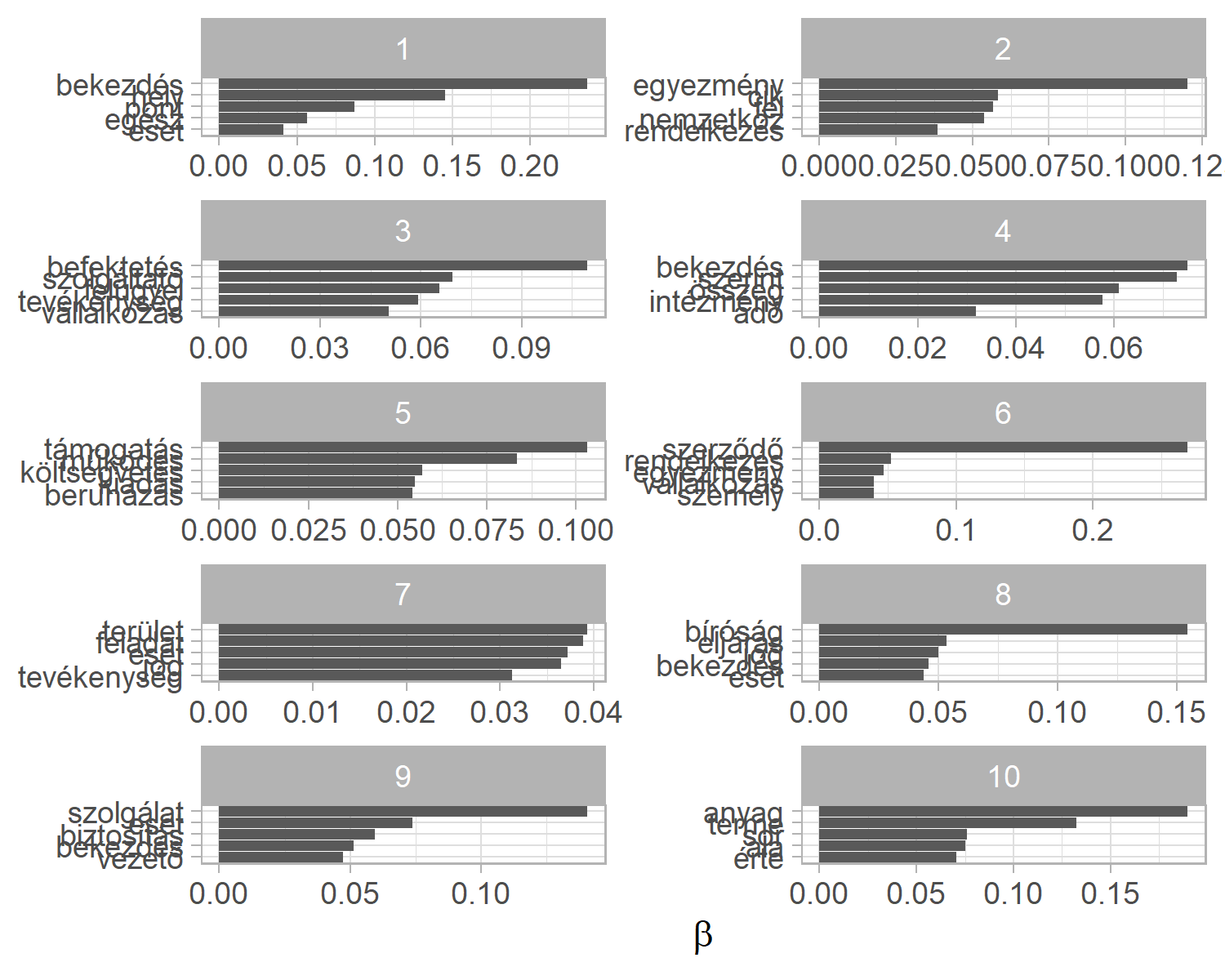


Figure 7.4: 1998-2002 ciklus topikok és kifejezések (VEM mintavételezéssel)

top\_terms %>%  
 filter(electoral\_cycle == "2002-2006") %>%  
 ggplot(aes(reorder\_within(term, beta, topic), beta)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~topic, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(  
 x = NULL,  
 y = expression(beta)  
 ) +  
 tidytext::scale\_x\_reordered()

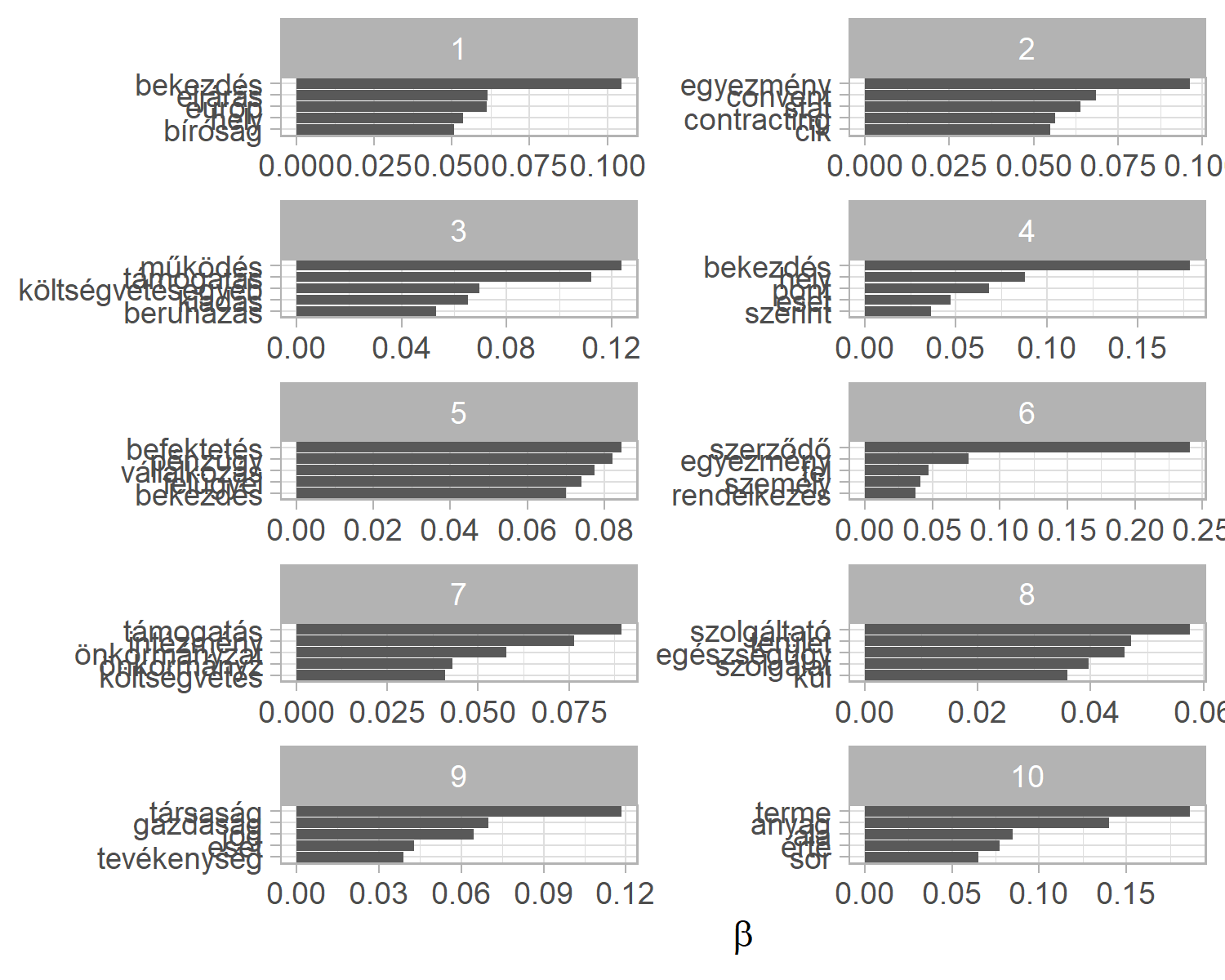


Figure 7.5: 2002-2006 ciklus topikok és kifejezések (VEM mintavételezéssel)

### 7.2.2 Az „LDA Gibbs" módszer alkalmazása a magyar törvények korpuszán

A következőkben ugyanazon a korpuszon az LDA Gibbs módszert alkalmazzuk. A szövegelőkészítő és tisztító lépések ennél a módszernél is ugyanazok mint a fentebb bemutatott “VEM” módszer esetében, így itt most csak a modell illesztését mutatjuk be.

gibbs\_98\_02 <- LDA(dfm\_98\_02, k = 10, method = "Gibbs", control = list(seed = 1234))  
  
gibbs\_02\_06 <- LDA(dfm\_02\_06, k = 10, method = "Gibbs", control = list(seed = 1234))

Itt is elvégezzük a topikok tidy formátumra alakítását.

topics\_g98\_02 <- tidy(gibbs\_98\_02, matrix = "beta") %>%  
 mutate(electoral\_cycle = "1998-2002")  
  
topics\_g02\_06 <- tidy(gibbs\_02\_06, matrix = "beta") %>%  
 mutate(electoral\_cycle = "2002-2006")  
  
lda\_gibbs <- bind\_rows(topics\_g98\_02, topics\_g02\_06)

Majd listázzuk az egyes topikokhoz tartozó leggyakoribb kifejezéseket.

top\_terms\_gibbs <- lda\_gibbs %>%  
 group\_by(electoral\_cycle, topic) %>%  
 top\_n(5, beta) %>%  
 top\_n(5, term) %>%  
 ungroup() %>%  
 arrange(topic, -beta)

Majd a ggplot2 csomag segítségével ábrán is megjeleníthetjük.

top\_terms\_gibbs %>%  
 filter(electoral\_cycle == "1998-2002") %>%  
 ggplot(aes(reorder\_within(term, beta, topic), beta)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~topic, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(  
 title = ,  
 x = NULL,  
 y = expression(beta)  
 ) +  
 tidytext::scale\_x\_reordered()

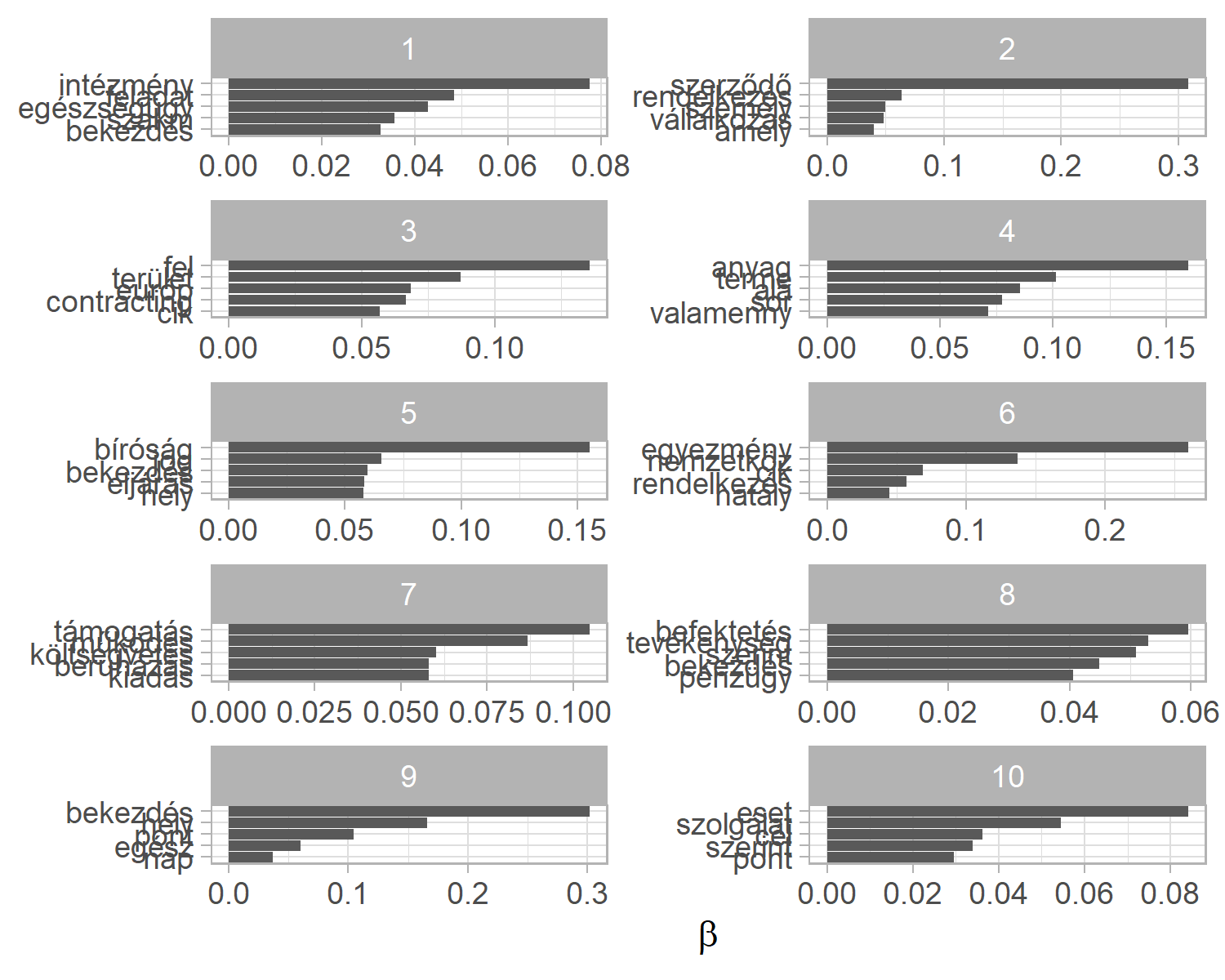


Figure 7.6: 1998-2002 ciklus topikok és kifejezések (Gibbs mintavétellel)

top\_terms\_gibbs %>%  
 filter(electoral\_cycle == "2002-2006") %>%  
 ggplot(aes(reorder\_within(term, beta, topic), beta)) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~topic, scales = "free", ncol = 2) +  
 coord\_flip() +  
 labs(  
 x = NULL,  
 y = expression(beta)  
 ) +  
 scale\_x\_reordered()

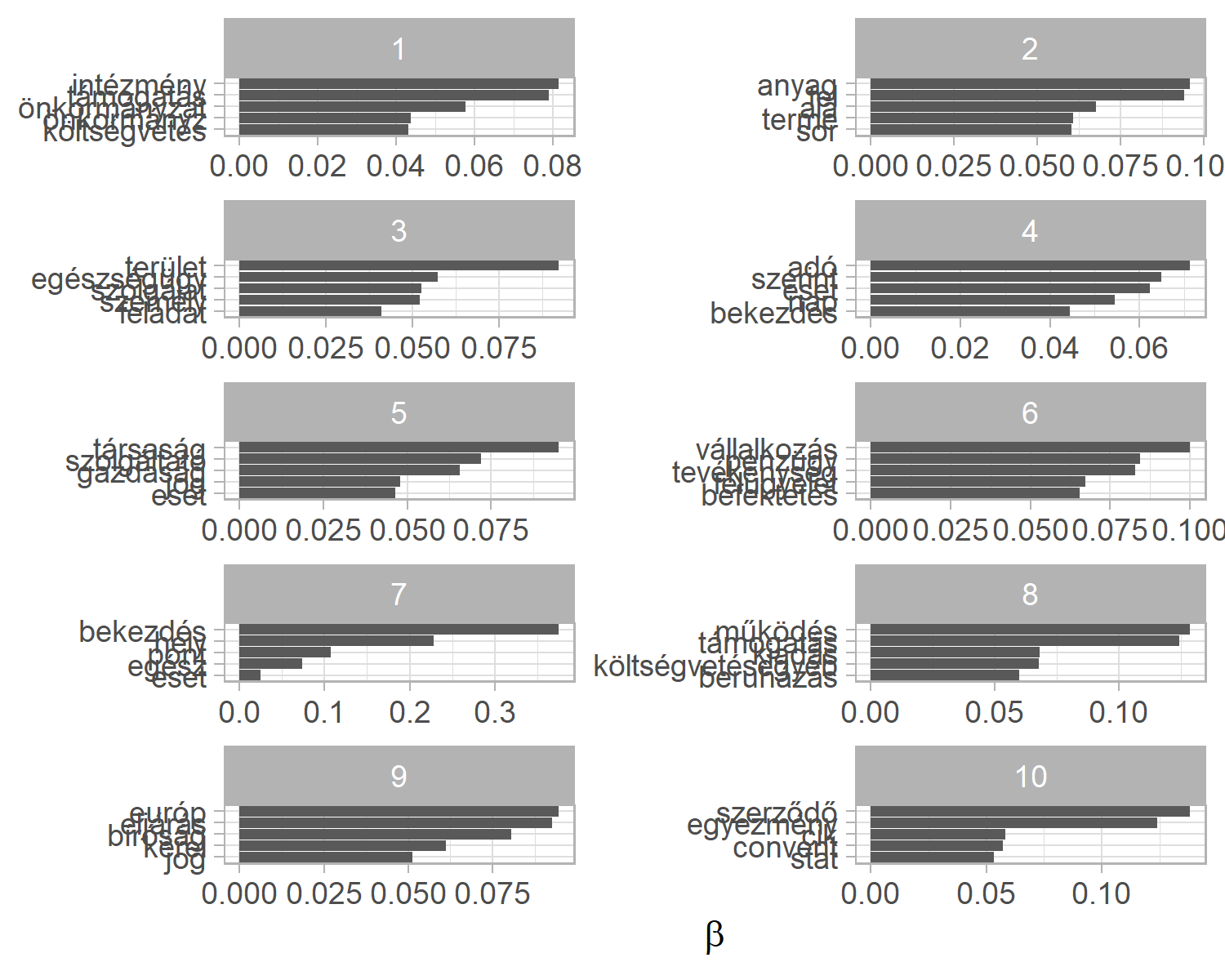


Figure 7.7: 2002-2006 ciklus topikok és kifejezések (Gibbs mintavétellel)

## 7.3 Struktúrális topik modellek

A kvantitatív szövegelemzés elterjedésével együtt megjelentek a módszertani innovációk is. [Roberts et al.](#ref-roberts2014structural) ([2014](#ref-roberts2014structural)) egy kiváló cikkben mutatta be a struktúrális topic modelleket (*structural topic models, stm*) amelyek fő újítása, hogy a dokumentumok metaadatai kovariánsként[[32]](#footnote-152) tudják befolyásolni, hogy egy-egy kifejezés mekkora valószínűséggel lesz egy-egy téma része. A kovariánsok egyrészről megmagyarázhatják hogy egy-egy dokumentum mennyire függ össze egy-egy témával (*topical prevalence*), illetve hogy egy-egy szó mennyire függ össze egy-egy témán belül (*topical content*).

Az stm modell becslése során mindkét típusú kovariánst használhatjuk, illetve hogyha nem adunk meg dokumentum meta adatot akkor az stm csomag stm függvénye a **Korrelált Topic Modell**-t fogja becsülni.

Az stm modelleket az R-ben az stm csomaggal tudjuk kivitelezni. A csomag fejlesztői között van a módszer kidolgozója is, ami nem ritka az R csomagok esetében.

A lenti lépésekben a csomag dokumentációjában szereplő ajánlásokat követjük, habár a könyv írásakor a stm már képes volt a quanteda-ban létrehozott dfm-ek kezelésére is. A kiinduló adatbázisunk a törvény\_final amit a fejezet elején hoztunk létre a dokumentumokból és a metaadatokból. A javasolt munkafolyamat a textProcessor()-használatával indul, ami szintén tartalmazza az alap szöveg előkészítési lépéseket. Az egyszerűség és futási sebesség érdekében itt most ezek többségétől eltekintünk, mivel a fejezet korábbi részeiben részletesen tárgyaltuk őket.

Az előkészítés utolsó szakaszában az out objektumban tároljuk el a dokumentumokat, egyedi szavakat, illetve a meta adatokat (kovariánsokat).

data\_stm <- torveny\_final  
  
processed\_stm <- textProcessor(  
 torveny\_final$text,  
 metadata = torveny\_final,  
 lowercase = FALSE,  
 removestopwords = FALSE,  
 removenumbers = FALSE,  
 removepunctuation = FALSE,  
 ucp = FALSE,  
 stem = TRUE,  
 language = "hungarian",  
 verbose = FALSE  
)  
  
out <- prepDocuments(processed\_stm$documents, processed\_stm$vocab, processed\_stm$meta)

A struktúrális topic modellünket az stm függvénnyel becsüljük és a kovariánsokat a prevalence opciónál tudjuk formulaként megadni. A lenti példában a Hungarian Comparative Agendas Projekt[[33]](#footnote-153) kategóriáit (pl.: gazdaság, egészségügy, stb.) és a kormányciklusokat használjuk. A futási idő kicsit hosszabb mint az LDA modellek esetében.

stm\_fit <- stm(  
 out$documents,  
 out$vocab,  
 K = 10,  
 prevalence = ~ majortopic + electoral\_cycle,  
 data = out$meta,  
 init.type = "Spectral",  
 seed = 1234,  
 verbose = FALSE  
)

Amennyiben a kutatási kérdés megkívánja, akkor megvizsálhatjuk hogy a kategórikus változóinknak milyen hatása volt egyes topikok esetében. Ehhez az estimateEffect() függvénnyel lefuttatunk egy lineáris regressziót és a summary() használatával láthatjuk az egyes kovariánsok koefficienseit. Itt az első topikkal illusztráljuk az eredményt, ami azt mutatja hogy (a kategórikus változóink első kategoriájához mérten) statisztikailag szignifikáns mint a téma mind pedig a kormányzati ciklusok abban hogy egyes dokumentumok milyen témákból épülnek fel.

out$meta$electoral\_cycle <- as.factor(out$meta$electoral\_cycle)  
out$meta$majortopic <- as.factor(out$meta$majortopic)  
  
cov\_estimate <- estimateEffect(1:10 ~ majortopic + electoral\_cycle, stm\_fit, meta = out$meta,   
 uncertainty = "Global")  
  
summary(cov\_estimate, topics = 1)  
#>   
#> Call:  
#> estimateEffect(formula = 1:10 ~ majortopic + electoral\_cycle,   
#> stmobj = stm\_fit, metadata = out$meta, uncertainty = "Global")  
#>   
#>   
#> Topic 1:  
#>   
#> Coefficients:  
#> Estimate Std. Error t value Pr(>|t|)   
#> (Intercept) 0.30118 0.03083 9.770 < 2e-16 \*\*\*  
#> majortopic2 -0.20953 0.06474 -3.237 0.001249 \*\*   
#> majortopic3 -0.19942 0.05950 -3.352 0.000833 \*\*\*  
#> majortopic4 -0.21489 0.05733 -3.749 0.000188 \*\*\*  
#> majortopic5 0.10192 0.04651 2.191 0.028674 \*   
#> majortopic6 -0.21813 0.05763 -3.785 0.000163 \*\*\*  
#> majortopic7 -0.15750 0.06746 -2.335 0.019745 \*   
#> majortopic8 -0.20311 0.07402 -2.744 0.006177 \*\*   
#> majortopic9 0.49074 0.10776 4.554 5.91e-06 \*\*\*  
#> majortopic10 -0.11412 0.05360 -2.129 0.033508 \*   
#> majortopic12 -0.17535 0.04068 -4.311 1.79e-05 \*\*\*  
#> majortopic13 -0.14383 0.05538 -2.597 0.009540 \*\*   
#> majortopic14 -0.21523 0.07339 -2.933 0.003435 \*\*   
#> majortopic15 -0.14833 0.04171 -3.556 0.000393 \*\*\*  
#> majortopic16 -0.09580 0.05444 -1.760 0.078762 .   
#> majortopic17 -0.22235 0.05728 -3.882 0.000110 \*\*\*  
#> majortopic18 0.21233 0.05565 3.816 0.000144 \*\*\*  
#> majortopic19 0.06358 0.04853 1.310 0.190472   
#> majortopic20 -0.21211 0.03944 -5.378 9.37e-08 \*\*\*  
#> majortopic21 -0.21810 0.06868 -3.176 0.001541 \*\*   
#> majortopic23 -0.19028 0.09082 -2.095 0.036404 \*   
#> electoral\_cycle2002-2006 -0.10232 0.01946 -5.257 1.78e-07 \*\*\*  
#> ---  
#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Az LDA modelleknél már bemutatott munkafolyamat az stm modellünk esetében is alkalmazható, hogy vizuálisan is megjelenítsük az eredményeinket. A tidy() data frammé alakítja az stm objektumot, amit aztán a már ismerős dplyr csomagban lévő függvényekkel tudunk átalakítani és végül vizualizálni a ggplot2 csomaggal. A lenti ábrán az egyes témákhoz tartozó 5 legvalószínűbb szót mutatjuk be.

tidy\_stm <- tidy(stm\_fit)

tidy\_stm %>%  
 group\_by(topic) %>%  
 top\_n(5, beta) %>%  
 ungroup() %>%  
 mutate(  
 topic = paste0("Topic ", topic),  
 term = reorder\_within(term, beta, topic)  
 ) %>%  
 ggplot(aes(term, beta)) +  
 geom\_col() +  
 facet\_wrap(~topic, scales = "free\_y", ncol = 3) +  
 coord\_flip() +  
 scale\_x\_reordered() +  
 labs(  
 x = NULL,  
 y = expression(beta)  
 )

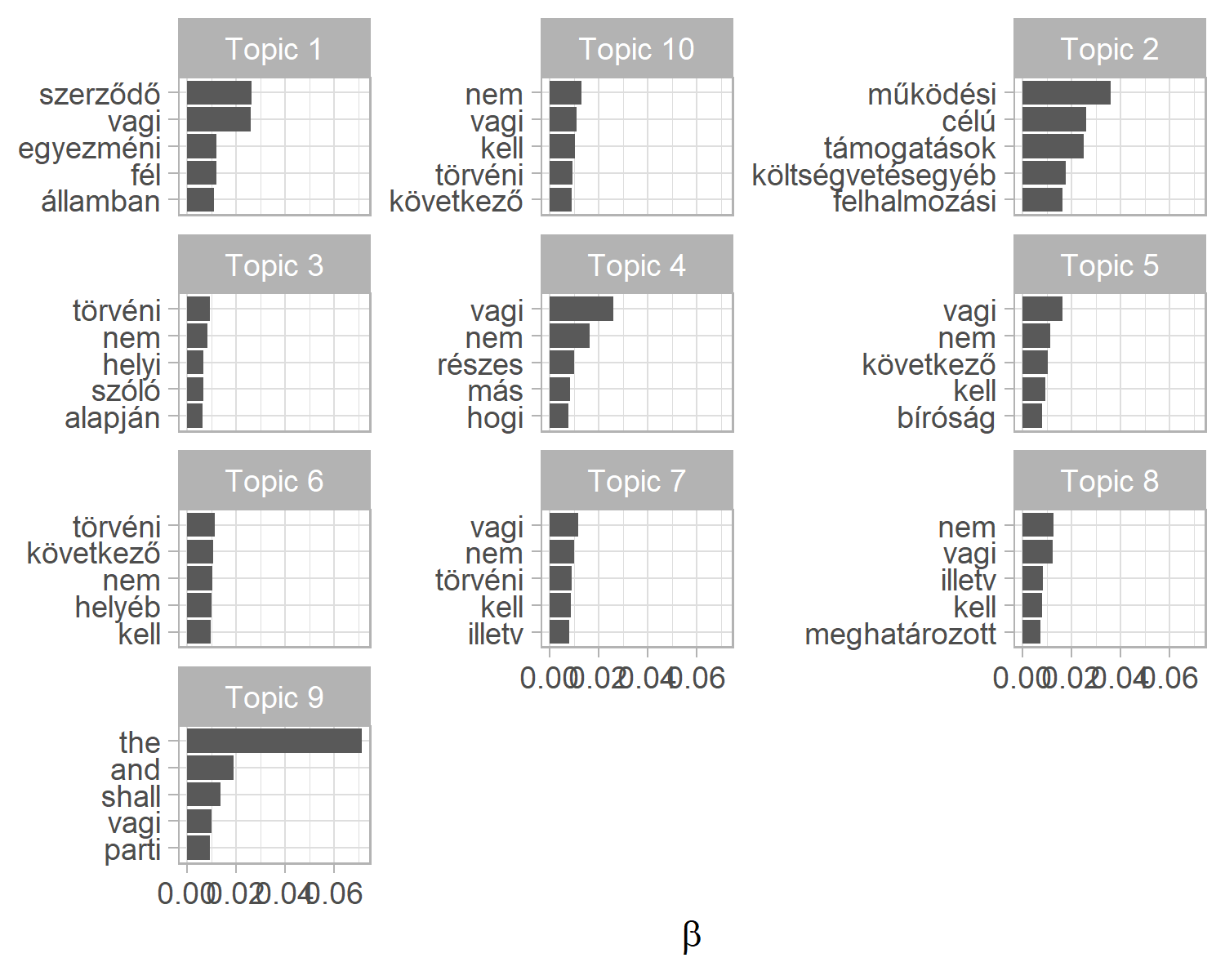


Figure 7.8: Topikonkénti legmagasabb valószínuségu szavak

Egy-egy topichoz tartozó meghatározó szavak annak függvényében változhatnak hogy milyen algoritmust használunk. A labelTopics() a már becsült stm modellünket alapul véve kínál 4 féle alternatív opciót. Az egyes algoritmusok részletes magyarázatáért érdemes elolvasni a csomag részletes leírását.[[34]](#footnote-155)

labelTopics(stm\_fit, c(1:2))  
#> Topic 1 Top Words:  
#> Highest Prob: szerzodo, vagi, egyezméni, fél, államban, nem, másik   
#> FREX: megadóztatható, haszonhúzója, beruházóinak, segélycsapatok, adóztatást, jövedelemadók, kijelölések   
#> Lift: árucikkeket, átalányösszegben, átléphetik, átszállítást, beruházóikat, célországban, cikktanulók   
#> Score: szerzodo, államban, illetoségu, egyezméni, megadóztatható, adóztatható, cikka   
#> Topic 2 Top Words:  
#> Highest Prob: muködési, célú, támogatások, költségvetésegyéb, felhalmozási, terhelo, beruházási   
#> FREX: kiadásokfelújításegyéb, kiadásokintézményi, kiadásokközponti, költségvetésfelhalmozási, kiadásokkormányzati, felújításegyéb, rek   
#> Lift: a+b+c, a+b+c+d, adago, adódóa, adósságállományából, adósságrendezésr, adótartozásának   
#> Score: költségvetésegyéb, költségvetésszemélyi, kiadásokfelhalmozási, járulékokdolog, költségvetésintézményi, kiadásokegyéb, juttatásokmunkaadókat

A korpuszunkon belüli témák megoszlását a plot.STM()-el tudjuk ábrázolni. Jól látszik hogy a Topic 6-ba tartozó szavak vannak jelen a legnagyobb arányban a dokumentumaink között.

plot.STM(stm\_fit, "summary", main = "", labeltype = "frex", xlab = "Várható topic arányok")

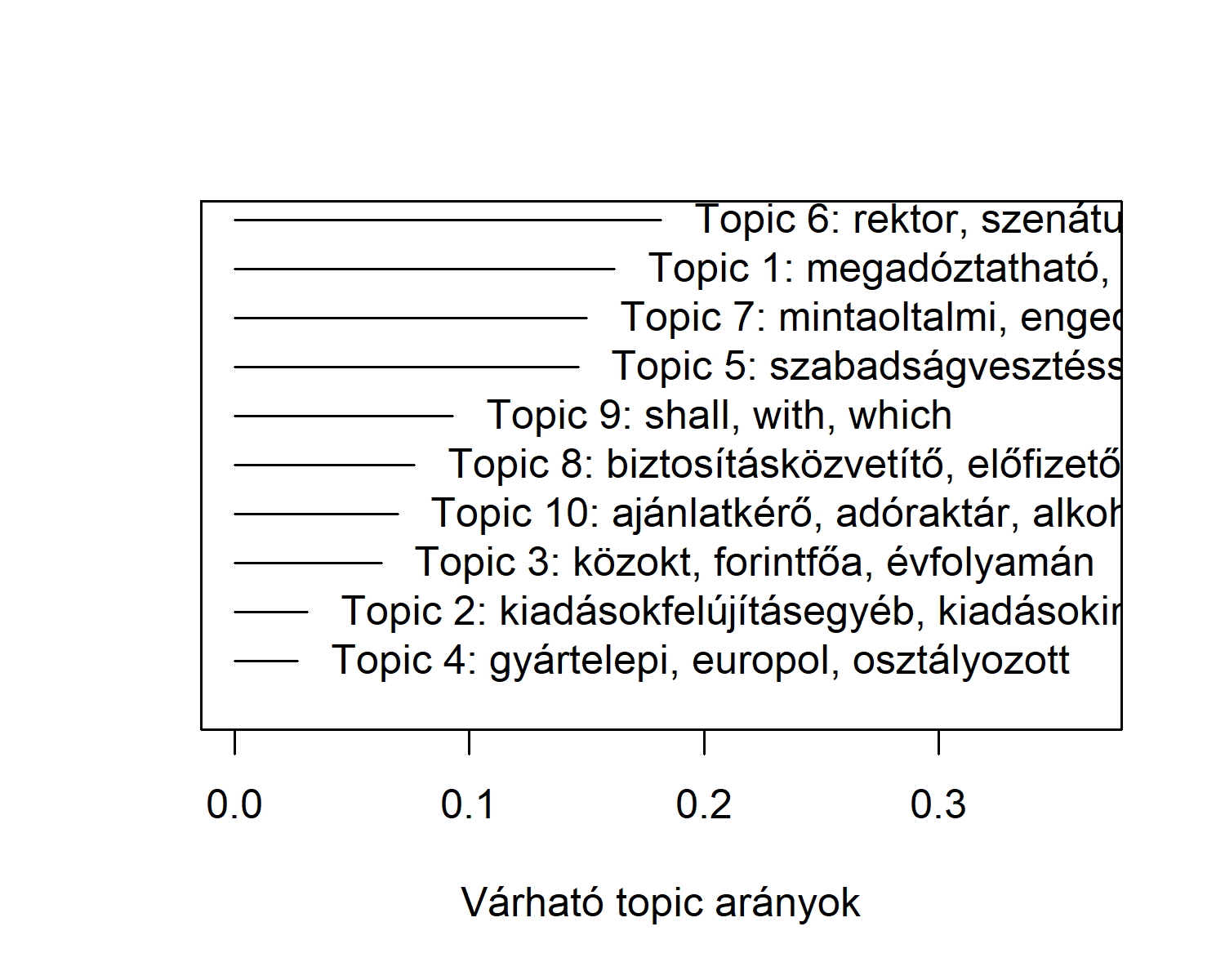


Figure 7.9: Leggyakoribb témák és kifejezések

Végezetül, a témák közötti korrelációt a topicCorr függvénnyel becsülhetjük és az igraph csomagot betöltve a plot() paranccsal tudjuk vizualizálni. Az eredmény egy hálózat lesz amit gráfként ábrázolunk. Az élei a gráfoknak a témák közötti összefüggést (korrelációt) jelölik.

plot(topicCorr(stm\_fit))

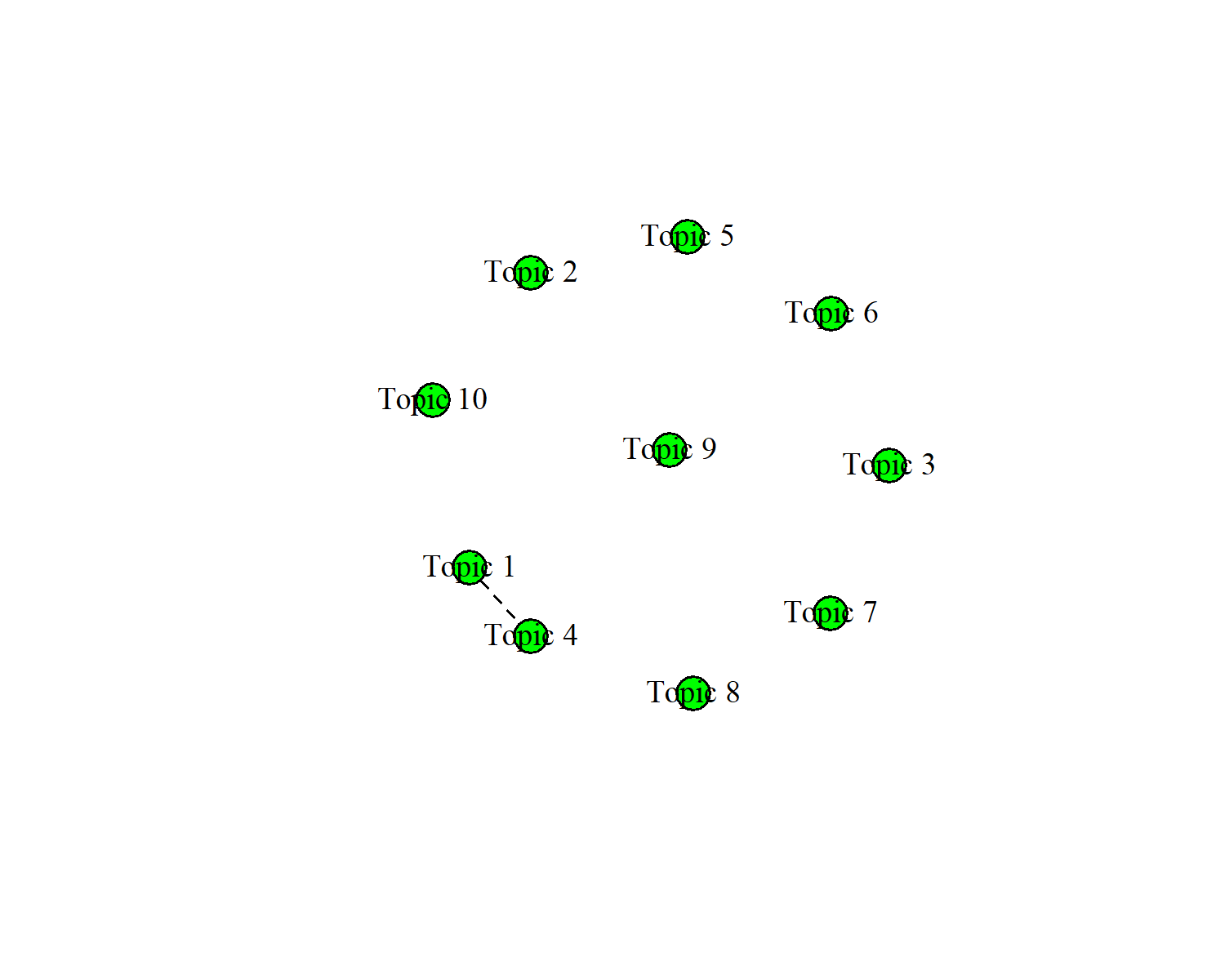


Figure 7.10: Témák közötti korreláció hálózat

# 8 Szóbeágyazások

Az eddigi fejezetekben elsősorban a szózsák (*bag of words*) alapú módszerek voltak előtérben. A szózsák alapú módszerekkel szemben, amelyek alkalmazása során elveszik a kontextuális tartalom, a szóbeágyazáson (*word embedding*) alapuló modellek kimondottan a kontextuális információt ragadják meg. A szóbeágyazás a topikmodellekhez hasonlóan a felügyelet nélküli tanulás módszerére épül, azonban itt a dokumentum domináns kifejezéseinek és témáinak feltárása helyett a szavak közötti szemantikai kapcsolat megértése a cél. Vagyis a modellnek képesnek kell lennie az egyes szavak esetén szinonimáik és ellentétpárjaik megtalálására.

A hagyományos topikmodellezés esetén a modell a szavak dokumentumokon belüli együttes megjelenési statisztikái alapján becsül dokumentum-topik, illetve topik-szó eloszlásokat, azzal a céllal, hogy koherens téma-csoportokat képezzen, ezzel szemben a szóbeágyazás legújabb iskolája már neurális halókon alapul. A neurális háló a tanítási folyamata során az egyes szavak vektorreprezentációját állítja elő. A vektorok jellemzően 100-300 dimenzióból állnak, a távolságuk alapján pedig megállapítható, hogy az egyes kifejezések milyen szemantikai kapcsolatban állnak egymással.

A szóbeágyazás célja tehát a szemantikai relációk feltárása. A szavak vektorizálásának köszönhetően bármely (a korpuszunkban szereplő) tetszőleges számú szóról eldönthetjük, hogy azok milyen szemantikai kapcsolatban állnak egymással – szinonimaként, vagy ellentétes fogalompárként szerepelnek. A szóvektorokon dimenziócsökkentő eljárást alkalmazva, s a multidimenzionális (100-300 dimenziós) teret 2 dimenziósra szűkítve könnyen vizualizálhatjuk is a korpuszunk kifejezései között fennálló szemantikai távolságot, és ahogy a lenti ábrákon is, láthatjuk, hogy az egyes kifejezések milyen relációban állnak egymással – a szemantikailag hasanló tartalmú kifejezések egymáshoz közel, míg a távolabbi jelentéstartalmú kifejezések egymástól távolabb foglalnak helyet. A klasszikus példa, amivel jól lehet szemléltetni a szóvektorok közötti összefüggést: king - man + woman = queen

## 8.1 Word2Vec, GloVe és fastText

A szóbeágyazásra társadalomtudományokban a két legnépszerűbb algoritmus – Word2Vec és a GloVe – a kontextuális szövegeloszláson (*distributional similarity based representations*) alapul, vagyis abból a feltevésből indul ki, hogy a hasonló kifejezések hasonló kontextusban fordulnak elő, emellett mindkettő sekély neurális hálón (2 rejtett réteg) alapuló modell.[[35]](#footnote-161) A Word2Vec-nek két verziója van: *Continuous Bag-of-words* (CBOW) és *SkipGram* (SG) – előbbi a kontextuális szavakból jelzi előre (*predicting*) a kontextushoz legszorosabban kapcsolódó kifejezést, míg utóbbi adott kifejezésből jelzi előre a kontextust [Mikolov et al.](#ref-mikolov2013efficient) ([2013](#ref-mikolov2013efficient)). A GloVe (*Global Vectors for Word Representation*) a Word2Vec-hez hasonlóan neurális hálón alapuló, szóvektorok előállítását célzó modell, a Word2Vec-kel szemben azonban nem a meghatározott kontextus-ablakban (*context window*) megjelenő kifejezések közti kapcsolatokat tárja fel, hanem a szöveg globális jellemzőit igyekszik megragadni az egész szöveget jellemző együttes előfordulási gyakoriságok (*co-occurrance*) meghatározásával [Pennington, Socher, and Manning](#ref-pennington2014glove) ([2014](#ref-pennington2014glove)). Míg a Word2Vec modell prediktív jellegű, addig a GloVe egy statisztikai alapú (*count-based*) modell, melyek gyakorlati hasznosításukat tekintve nagyon hasonlóak.

A szóvektor modellek között érdemes megemlíteni a fastText-et is, mely 157 nyelvre kínál (köztük magyarra is) a szóbeágyazás módszeren alapuló, előre tanított szóvektorokat, melyet tovább lehet tanítani speciális szövegkorpuszokra, ezzel jelentősen lerövidítve a modell tanításához szükséges idő-, és kapacitásszükségletet [Mikolov et al.](#ref-mikolov2018advances) ([2018](#ref-mikolov2018advances)). Habár a GloVe és Word2Vec skip-gram módszerek hasonlóságát a szakirodalom adottnak veszi, a tényleges kép ennél árnyaltabb. A GloVe esetében a ritkán előforduló szavak kisebb súlyt kapnak a szóvektorok számításánál, míg a Word2Vec alulsúlyozza a nagy frekvenciájú szavakat. Ennek a következménye, hogy a Word2Vec esetében gyakori hogy a szemantikailag legközelebbi szó az egy elütés, nem pedig valid találat. Ennek ellenére a két módszer (amennyiben a Word2Vec algoritmusnál a kisfrekvenciájú tokeneket kiszűrjük) az emberi validálás során nagyon hasonló eredményeket hozott ([Spirling and Rodriguez Forthcoming](#ref-spirlingword)).

A fejezetben a gyakorlati példa során a GloVe algoritmust használjuk majd, mivel véleményünk szerint az implementációt tartalmazó R csomagnak jobb és könnyebben követhető a dokumentációja mint a többi alternatívának.

### 8.1.1 GloVe használata magyar média korpuszon

Az elemzéshez a text2vec csomagot használjuk, ami a GloVe implementációt tartalmazza. A lenti kód a csomag dokumentáción alapul és a Társadalomtudományi Kutatóközpont által a *Hungarian Comparative Agendas Project (CAP)* adatbázisában tárolt Magyar Nemzet korpuszt használja.[[36]](#footnote-162)

library(text2vec)  
library(quanteda)  
library(readtext)  
library(readr)  
library(dplyr)  
library(tibble)  
library(stringr)  
library(ggplot2)

A lenti kód blokk azt mutatja be, hogy hogyan kell a betöltött korpuszt tokenizálni és mátrix formátumba alakítani. A korpusz az a Magyar Nemzet 2004 és 2014 közötti címlapos cikkeit tartalmazza. Az eddigi előkészítő lépéseket most is megtesszük: kitöröljük a központozást, számokat, magyar töltelékszavakat, illetve kisbetűsítünk és eltávolítjuk a felesleges szóközöket és tördeléseket.

mn <- read\_csv("data/mn\_large.csv")  
  
mn\_clean <- mn %>%  
 mutate(  
 text = str\_remove\_all(string = text, pattern = "[:cntrl:]"),  
 text = str\_remove\_all(string = text, pattern = "[:punct:]"),  
 text = str\_remove\_all(string = text, pattern = "[:digit:]"),  
 text = str\_to\_lower(text),  
 text = str\_trim(text),  
 text = str\_squish(text)  
 )

Fontos különbség hogy az eddigi munkafolyamatokkal ellentétben a GloVe algoritmus nem egy dokumentum-kifejezés mátrixon dolgozik, hanem egy kifejezések együttes előfordulását tartalmazó mátrixot (*feature co-occurence matrix*) kell készíteni inputként. Ezt a quanteda fcm() függvényével tudjuk előállítani, ami a tokenekből készíti el a mátrixot. A tokenek sorrendiségét úgy tudjuk megőrizni, hogy egy dfm objektumból csak a kifejezéseket tartjuk meg a featnames() függvény segítségével, majd a teljes token halmazból a tokens\_select() függvénnyel kiválasztjuk őket.

mn\_corpus <- corpus(mn\_clean)  
  
mn\_tokens <- tokens(mn\_corpus) %>%  
 tokens\_remove(stopwords(language = "hungarian"))  
  
features <- dfm(mn\_tokens) %>%  
 dfm\_trim(min\_termfreq = 5) %>%  
 featnames()  
  
mn\_tokens <- tokens\_select(mn\_tokens, features, padding = TRUE)

Az fcm megalkotása során a célkifejezéstől való távolság függvényében súlyozzuk a tokeneket.

mn\_fcm <- fcm(mn\_tokens, context = "window", count = "weighted", weights = 1/(1:5),   
 tri = TRUE)

A tényleges szóbeágyazás a text2vec csomaggal történik. A GlobalVector egy új “környezetet” (*environment*) hoz létre. Itt adhatjuk meg az alapvető paramétereket. A rank a vektor dimenziót adja meg (az irodalomban a 300-500 dimenzió a megszokott). A többi paraméterrel is lehet kísérletezni, hogy mennyire változtatja meg a kapott szóbeágyazásokat. A fit\_transform pedig a tényleges becslést végzi. Itt az iterációk számát (a gépi tanulásos irodalomban *epoch*-nak is hívják a tanulási köröket) és a korai leállás (*early stopping*) kritériumát a convergence\_tol megadásával. Minél több dimenziót szeretnénk és minél több iterációt, annál tovább fog tartani a szóbeágyazás futtatása.

Az egyszerűség és gyorsaság miatt a lenti kód 10 körös tanulást ad meg, ami a relatíve kicsi Magyar Nemzet korpuszon ~3 perc alatt fut le.[[37]](#footnote-164) Természetesen minél nagyobb korpuszon, minél több iterációt futtatunk, annál pontosabb eredményt fogunk kapni. A text2vec csomag képes a számítások párhuzamosítására, így alapbeállításként a rendelkezésre álló összes CPU magot teljesen kihasználja a számításhoz. Ennek ellenére egy százezres, milliós korpusz esetén több óra is lehet a tanítás.

glove <- GlobalVectors$new(rank = 300, x\_max = 10, learning\_rate = 0.1)  
  
mn\_main <- glove$fit\_transform(mn\_fcm, n\_iter = 10, convergence\_tol = 0.01)

A végleges szóvektorokat a becslés során elkészült két mátrix összegeként kapjuk.

mn\_context <- glove$components  
  
mn\_word\_vectors <- mn\_main + t(mn\_context)

Az egyes szavakhoz legközelebb álló szavakat a koszinusz hasonlóság alapján kapjuk, a sim2() függvénnyel. A lenti példában “l2” normalizálást alkalmazunk, majd a kapott hasonlósági vektort csökkenő sorrendbe rendezzük. Példaként a “polgármester” szónak a környezetét nézzük meg. Mivel a korpuszunk egy politikai napilap, ezért nem meglepő, hogy a legközelebbi szavak a politikához kapcsolódnak.

teszt <- mn\_word\_vectors["polgármester", , drop = F]  
  
cos\_sim\_rom <- sim2(x = mn\_word\_vectors, y = teszt, method = "cosine", norm = "l2")  
  
head(sort(cos\_sim\_rom[, 1], decreasing = TRUE), 5)  
#> polgármester mszps szocialista fideszes elmondta   
#> 1.0000000 0.5059529 0.4339177 0.4204766 0.4024232

A lenti show\_vector() függvényt definiálva a kapott eredmény egy data frame lesz, és az n változtatásával a kapcsolódó szavak számát is könnyen változtathatjuk.

show\_vector <- function(vectors, pattern, n = 5) {  
 term <- mn\_word\_vectors[pattern, , drop = F]  
 cos\_sim <- sim2(x = vectors, y = term, method = "cosine", norm = "l2")  
 cos\_sim\_head <- head(sort(cos\_sim[, 1], decreasing = TRUE), n)  
 output <- enframe(cos\_sim\_head, name = "term", value = "dist")  
 return(output)  
}

Példaként a “barack” nem gyümölcsöket fog adni, hanem az Egyesült Államok elnökét és hozzá kapcsolódó szavakat.

show\_vector(mn\_word\_vectors, "barack", 10)  
#> # A tibble: 10 x 2  
#> term dist  
#> <chr> <dbl>  
#> 1 barack 1.00   
#> 2 obama 0.691  
#> 3 elnök 0.372  
#> 4 amerikai 0.349  
#> 5 demokrata 0.339  
#> 6 republikánus 0.294  
#> 7 részesülhessenek 0.256  
#> 8 egyesült 0.253  
#> 9 elnököt 0.251  
#> 10 bush 0.239

Ugyanez működik magyar vezetőkkel is.

show\_vector(mn\_word\_vectors, "orbán", 10)  
#> # A tibble: 10 x 2  
#> term dist  
#> <chr> <dbl>  
#> 1 orbán 1.00   
#> 2 viktor 0.937  
#> 3 miniszterelnök 0.743  
#> 4 mondta 0.701  
#> 5 jelentette 0.673  
#> 6 kormányfo 0.667  
#> 7 fogalmazott 0.661  
#> 8 fidesz 0.656  
#> 9 hangsúlyozta 0.655  
#> 10 beszélt 0.624

Az irodalomban klasszikus vektorműveletes példákat is reprokuálni tudjuk a Magyar Nemzet korpuszon készített szóbeágyazásainkkal. A budapest - magyarország + német + németország eredményét úgy kapjuk meg, hogy az egyes szavakhoz tartozó vektorokat kivonjuk egymásból, illetve hozzáadjuk őket, ezután pedig a kapott mátrixon a quanteda csomag textstat\_simil függvényével kiszámítjuk hogy az új hasonlósági értékeket.

budapest <- mn\_word\_vectors["budapest", , drop = FALSE] - mn\_word\_vectors["magyarország", , drop = FALSE] + mn\_word\_vectors["német", , drop = FALSE] +  
 + mn\_word\_vectors["németország", , drop = FALSE]  
  
cos\_sim <- textstat\_simil(x = as.dfm(mn\_word\_vectors), y = as.dfm(budapest), method = "cosine")  
  
head(sort(cos\_sim[, 1], decreasing = TRUE), 5)  
#> budapest német németország kancellár angéla   
#> 0.6393528 0.6014288 0.5315928 0.4602480 0.4222456

A szavak egymastól való távolságát vizuálisan is tudjuk ábrázolni. Az egyik ezzel kapcsolatban felmerülő probléma, hogy egy 2 dimenziós ábrán akarunk egy 3-500 dimenziós mátrixot ábrázolni. Több lehetséges megoldás is van, mi ezek közül a lehető legegyszerűbbet mutatjuk be.[[38]](#footnote-165) Első lépésben egy data framet készítünk a szóbeágyazás eredményeként kapott mátrixból, megtartva a szavak az első oszlopban a tibble csomag rownames\_to\_column függvényével. Mivel csak 2 dimenziót tudunk ábrázolni egy tradíciónális statikus ábrán, ezért a V1 és V2 oszlopokat tartjuk csak meg, amik az első és második dimenziót reprezentálják.

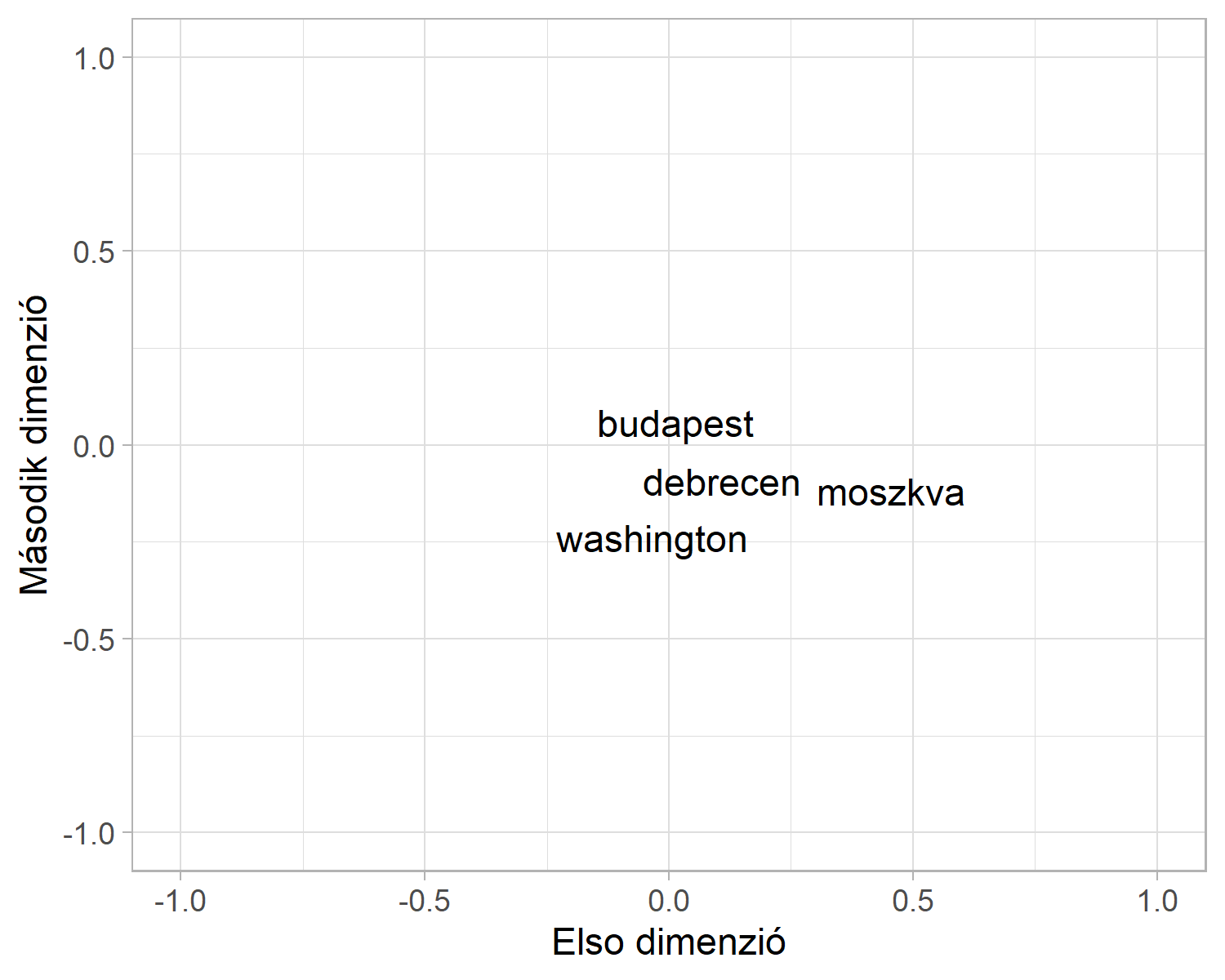
mn\_embedding\_df <- as.data.frame(mn\_word\_vectors[, c(1:2)]) %>%   
 rownames\_to\_column(var = "words")

Ezután pedig a ggplot függvényt felhasználva definiálunk egy új, embedding\_plot nevű, függvényt, ami az elkészült data frame alapján bármilyen kulcsszó kombinációt képes ábrázolni.

embedding\_plot <- function(data, keywords) {  
 data %>%   
 filter(words %in% keywords) %>%   
 ggplot(aes(V1, V2, label = words)) +  
 labs(  
 x = "Első dimenzió",  
 y = "Második dimenzió"  
 ) +  
 geom\_text() +  
 xlim(-1, 1) +  
 ylim(-1, 1)  
}

Példaként pár településnevet megvizsgálva azt látjuk hogy a külföldi fővárosok közel helyezkednek el egymástól, míg a magyar települések kissé távolabb. Ennek az lehet az oka, hogy a külföldi fővárosok inkább a külpolitikai cikkekben szerepelnek, míg a magyarok sokkal több kontextusban előkerülhetnek.

words\_selected <- c("moszkva", "debrecen", "budapest", "washington")  
  
embedding\_plot(data = mn\_embedding\_df, keywords = words\_selected)



# 9 Szövegskálázás: felügyelet nélküli és felügyelt megoldások

A szövegskálázás célja a politikai szereplők elhelyezése az ideológiai térben. Ennek felügyelt típusa a wordscores, amely a szótári módszerekhez hasonlóan a szereplőket szavaik alapján helyezi el a politikai térben, úgy hogy az ún.referencia dokumentumok szövegét használja tanító halmazként. A wordscores kiindulópontja, hogy pozíció pontszámokat kell rendelni referencia szövegekhez. A modell számításba veszi szövegek szavainak súlyozott gyakoriságát és a pozíciópontszám valamint a szógyakoriság alapján becsülni meg a korpuszban lévő többi dokumentum pozícióját. ([Laver, Benoit, and Garry 2003](#ref-laver2003extracting)). A felügyelet nélküli wordfish módszer a skálázás során nem a referencia dokumentumokra támaszkodik, hanem olyan kifejezéseket keres a szövegben, amelyek megkülönböztetik egymástól a politikai spektrum különböző pontjain elhelyezkedő beszélőket. Az IRT-n (item response theory) alapuló módszer azt feltételezi, hogy a politikusok egy kevés dimenziós politikai térben mozognak, amely tér leírható az *i* politikus paraméterével. Egy politikus (vagy párt) ezen a téren elfoglalt helyzete pedig befolyásolja a szavak szövegekben történő használatát. A módszer erőssége, hogy kevés erőforrás-befektetéssel megbízható becsléseket ad, ha a szövegek valóban az ideológiák mentén különböznek, tehát ha a szereplők erősen ideológiai tartalamú diskurzust folytatnak. Alkalmazásakor azonban tudnunk kell, hogy a módszer nem képes kezelni, hogy a szövegek között nem csak ideológiai különbség lehet. Mivel a modell nem felügyelt, ezért nehéz garantálni, hogy valóban megbízhatóan azonosítja a szereplők elhelyezkedését a politikai térben, így az eredményeket mindenképpen körültekintően kell validálni. ([Slapin and Proksch 2008](#ref-slapinScalingModelEstimating2008); [Hjorth et al. 2015](#ref-hjorthComputersCodersVoters2015); [Grimmer and Stewart 2013](#ref-grimmer2013text))

library(readr)  
library(dplyr)  
library(stringr)  
library(ggplot2)  
library(quanteda)  
library(quanteda.textmodels)

A skálázási algoritmusokat egy kis korpuszon mutatjuk be. A minta dokumentumok a 2014-2018 parlamenti ciklusban a Magyar Országgyűlésben frakcióvezető politikusok egy-egy véletlenszerűen kiválasztott napirend előtti felszólalásai. Ebben a ciklusban összesen 11 frakcióvezetője volt a két kormánypárti és öt ellenzéki frakciónak.[^skala-1] A dokumentumokon először elvégeztük a szokásos előkészítési lépéseket. [^skala-1]: A mintába nem került be Rogán Antal, akinek csak egy darab napirend előtti felszólalása volt.

parl\_beszedek <- read\_csv("data/ps\_sample.csv")  
  
beszedek\_tiszta <- parl\_beszedek %>%   
 mutate(  
 text = str\_remove\_all(string = text, pattern = "[:cntrl:]"),  
 text = str\_remove\_all(string = text, pattern = "[:punct:]"),  
 text = str\_remove\_all(string = text, pattern = "[:digit:]"),  
 text = str\_to\_lower(text),  
 text = str\_trim(text),  
 text = str\_squish(text)  
 )

A *Wordfish* és *Wordscores* algoritmus is ugyanazt a kiinduló corpus és dfm objektumot használja, amit a szokásos módon a quanteda csomag corpus() függvényével hozunk létre. A leíró statisztikai táblázatban látszik, hogy a beszédek hosszúsága nem egységes, a leghosszabb 10267 szavas, a legrövidebb pedig 1976. Az átlagos dokumentum hossz az 5136. A korpusz szemléltető célú, alaposabb elemzéshez hosszabb és/vagy több dokumentummal érdemes dolgoznunk.

beszedek\_corpus <- corpus(beszedek\_tiszta)  
  
summary(beszedek\_corpus)  
#> Corpus consisting of 10 documents, showing 10 documents:  
#>   
#> Text Types Tokens Sentences id  
#> text1 442 819 1 20142018\_024\_0002\_0002  
#> text2 354 607 1 20142018\_055\_0002\_0002  
#> text3 426 736 1 20142018\_064\_0002\_0002  
#> text4 314 538 1 20142018\_115\_0002\_0002  
#> text5 354 589 1 20142018\_158\_0002\_0002  
#> text6 333 538 1 20142018\_172\_0002\_0002  
#> text7 344 559 1 20142018\_206\_0002\_0002  
#> text8 352 628 1 20142018\_212\_0002\_0002  
#> text9 317 492 1 20142018\_236\_0002\_0002  
#> text10 343 600 1 20142018\_249\_0002\_0002  
#> felszolalo part  
#> Vona Gábor (Jobbik) Jobbik  
#> Dr. Schiffer András (LMP) LMP  
#> Dr. Szél Bernadett (LMP) LMP  
#> Tóbiás József (MSZP) MSZP  
#> Schmuck Erzsébet (LMP) LMP  
#> Dr. Tóth Bertalan (MSZP) MSZP  
#> Volner János (Jobbik) Jobbik  
#> Kósa Lajos (Fidesz) Fidesz  
#> Harrach Péter (KDNP) KDNP  
#> Dr. Gulyás Gergely (Fidesz) Fidesz

A koprusz létrehozása után elkészítjük a dfm mátrixot, amelyből eltávolítjuk a magyar stopszvakat a quantedabeépített szótárának segítségével.

beszedek\_dfm <- beszedek\_corpus %>%   
 tokens() %>%   
 tokens\_remove(stopwords("hungarian")) %>%   
 dfm()

## 9.1 Wordfish

A wordfish felügyelet nélküli skálázást a quanteda\_textmodels csomagban implementált textmodel\_wordfish() függvény fogja végezni. A megadott dir = c(1, 2) paraméterrel a két dokumentum relatív értékét tudjuk rögzíteni, mégpedig úgy hogy . Alapbeállításként az első és utolsó dokumentumot teszi ide be az algoritmus. A lenti példánál mi a pártpozíciók alapján a Jobbikos Vona Gábor és az LMP-s Schiffer András egy-egy beszédét használtuk. A summary() használható az illesztett modellel, és a dokumentumonkénti koefficienst tudjuk így megnézni.

beszedek\_wf <- textmodel\_wordfish(beszedek\_dfm, dir = c(2, 1))  
  
summary(beszedek\_wf)  
#>   
#> Call:  
#> textmodel\_wordfish.dfm(x = beszedek\_dfm, dir = c(2, 1))  
#>   
#> Estimated Document Positions:  
#> theta se  
#> text1 1.79474 0.04219  
#> text2 0.08931 0.04001  
#> text3 1.00137 0.03908  
#> text4 -0.09988 0.04232  
#> text5 0.73596 0.04355  
#> text6 0.18572 0.04452  
#> text7 -0.72832 0.03590  
#> text8 -0.80587 0.03358  
#> text9 -0.52028 0.04005  
#> text10 -1.65273 0.03794  
#>   
#> Estimated Feature Scores:  
#> vona gábor jobbik tisztelt elnök úr országgyulés tegnapi  
#> beta 3.675 2.321 1.9710 0.2391 -0.11149 0.02755 1.2286 4.372  
#> psi -4.980 -2.734 -0.7531 0.4566 -0.05693 0.28721 -0.6705 -5.314  
#> napon helyen tartottak idoközi önkormányzati választásokat két  
#> beta 2.991 3.103 3.675 3.675 3.675 3.675 1.1894  
#> psi -3.009 -2.630 -4.980 -4.980 -4.980 -4.980 -0.9439  
#> érdekelt recsken ózdon október nyertünk örömmel közlöm ország  
#> beta 3.675 4.372 4.774 3.405 3.675 3.675 3.675 1.7470  
#> psi -4.980 -5.314 -5.545 -3.230 -4.980 -4.980 -4.980 -0.3643  
#> közvéleményével amúgy is tudnak mindkét jobbikos polgármester  
#> beta 3.675 3.675 0.9128 1.433 3.675 3.675 3.675  
#> psi -4.980 -4.980 1.8345 -1.737 -4.980 -4.980 -4.980

Amennyiben szeretnénk a szavak szintjén is megnézni a (a szavakhoz társított súly, ami a relatív fontosságát mutatja) és (a szó fix effekt, ami az eltérő szófrekvencia kezeléséért felelős) koefficiensekhez, akkor a beszedek\_wf objektumban tárolt értékeket egy data frame-be tudjuk bemásolni. A dokumentumok hosszára és a szófrekvenciát figyelembe véve, a negatív értékű szavakat gyakrabban használják a negatív koefficienssel rendelkező politikusok.

szavak\_wf <- data.frame(  
 word = beszedek\_wf$features,   
 beta = beszedek\_wf$beta,   
 psi = beszedek\_wf$psi  
 )  
  
szavak\_wf %>%   
 arrange(beta) %>%   
 head(n = 15)  
#> word beta psi  
#> 1 czeglédy -5.900663 -6.222629  
#> 2 csaba -5.769959 -6.151399  
#> 3 human -5.438681 -5.975155  
#> 4 operator -5.438681 -5.975155  
#> 5 zrt -5.216835 -5.860931  
#> 6 fizette -4.927204 -5.717002  
#> 7 gyanú -4.927204 -5.717002  
#> 8 szocialista -4.927204 -5.717002  
#> 9 elkövetett -4.509192 -5.521276  
#> 10 tárgya -4.509192 -5.521276  
#> 11 céghálózat -4.509192 -5.521276  
#> 12 diákok -4.509192 -5.521276  
#> 13 májusi -4.509192 -5.521276  
#> 14 júniusi -4.509192 -5.521276  
#> 15 büntetoeljárás -4.509192 -5.521276

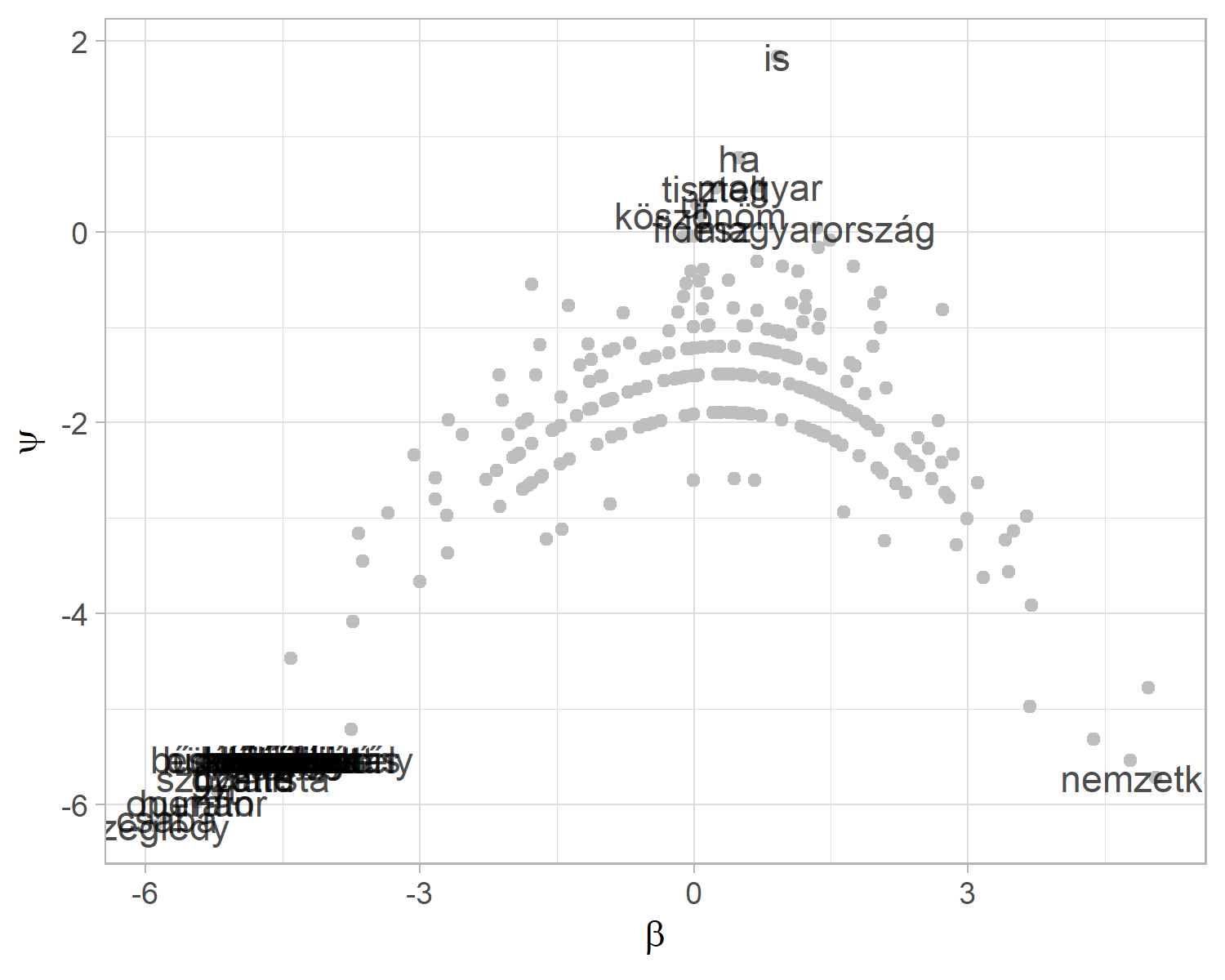
Ez a pozitív értékekre is igaz.

szavak\_wf %>%   
 arrange(desc(beta)) %>%   
 head(n = 15)  
#> word beta psi  
#> 1 nemzetközi 5.057078 -5.720709  
#> 2 önöknek 4.977502 -4.778607  
#> 3 ózdon 4.773523 -5.544626  
#> 4 kétharmados 4.773523 -5.544626  
#> 5 igenis 4.773523 -5.544626  
#> 6 választási 4.773523 -5.544626  
#> 7 geopolitikai 4.773523 -5.544626  
#> 8 ártatlanság 4.773523 -5.544626  
#> 9 vélelme 4.773523 -5.544626  
#> 10 tegnapi 4.372320 -5.314088  
#> 11 recsken 4.372320 -5.314088  
#> 12 lássuk 4.372320 -5.314088  
#> 13 tolünk 4.372320 -5.314088  
#> 14 janiczak 4.372320 -5.314088  
#> 15 szavazattal 4.372320 -5.314088

Az eredményeinket mind a szavak és mind a dokumentumok szintjén tudjuk vizualizálni. Elsőként a klasszikus “Eiffel-torony” ábrát reprodukáljuk, ami a szavak gyakorisága és skálára gyakorolt befolyásának az illusztrálására szolgál. Ehhez a már elkészült szavak\_wf data framet és a ggplot2 csomagot fogjuk használni. Mivel a korpuszunk nagyon kicsi ezért csak 2410 kifejezést fogunk ábrázolni. Ennek ellenére a lényeg kirajzolódik a lenti ábrán is.[[39]](#footnote-170)

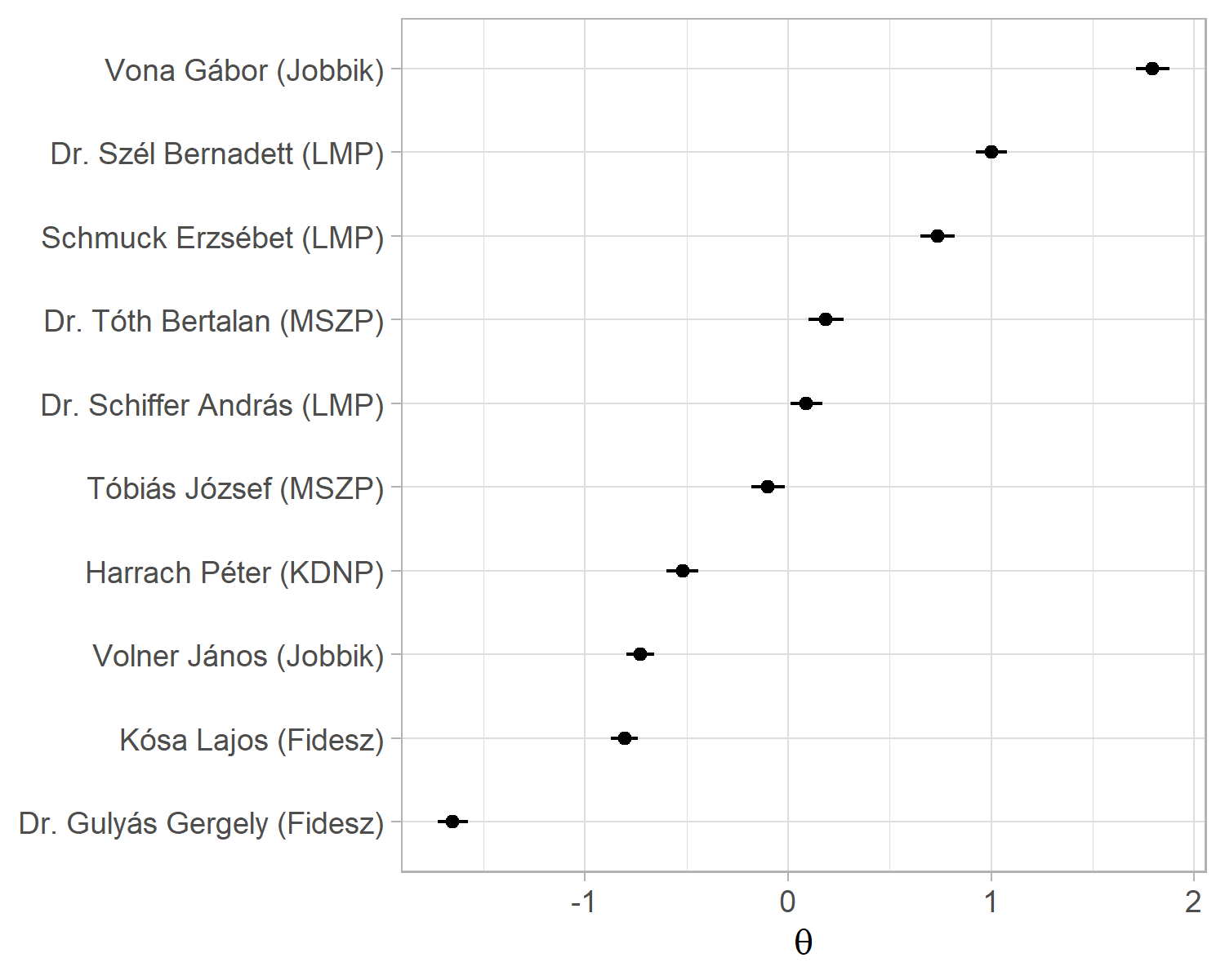
Kihasználhatjuk, hogy a ggplot ábra definiálása közben a felhasznált bemeneti data frame-t különböző szempontok alapján lehet szűrni. így ábrázolni tudjuk a gyakran használt ám semleges szavakat (magas , alacsony ), illetve a ritkább de meghatározóbb szavakat (magas , alacsony ).

ggplot(szavak\_wf, aes(x = beta, y = psi)) +  
 geom\_point(color = "grey") +  
 geom\_text(  
 data = filter(szavak\_wf, beta > 5 | beta < -4.5 | psi > 0),  
 aes(beta, psi, label = word),  
 alpha = 0.7  
 ) +  
 labs(  
 x = expression(beta),  
 y = expression(psi)  
 )



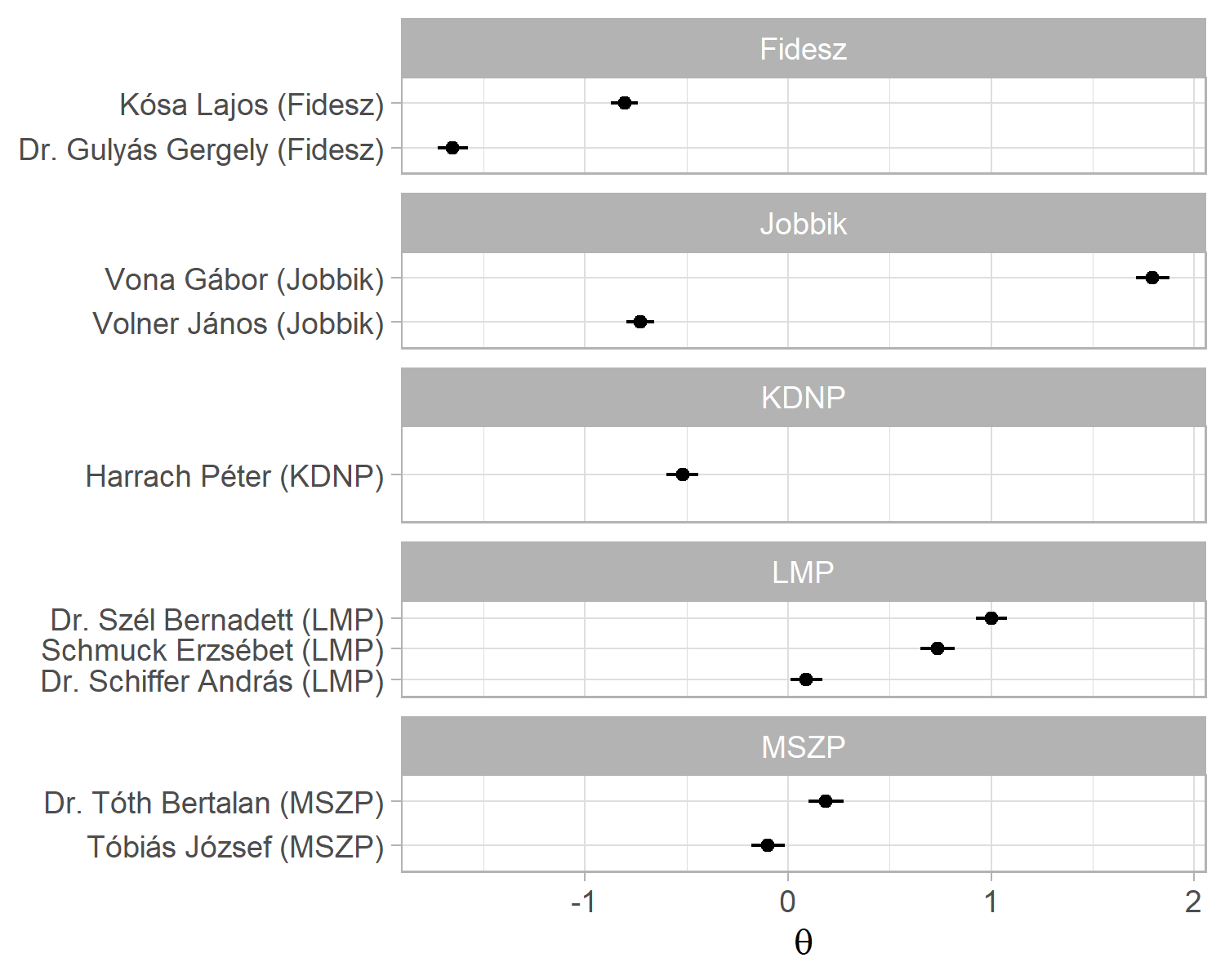
A dokumentumok szintjén is érdemes megvizsgálni az eredményeket. Ehhez a dokumentum szintű paramétereket fogjuk egy data framebe gyűjteni: a ideológiai pozíciót, illetve a beszélő nevét. A vizualizáció kedvéért a párttagságot is hozzáadjuk. A data frame összerakása után az alsó és felső határát is kiszámoljuk a konfidencia intervallumnak és azt is ábrázoljuk.

dokumentumok\_wf <- data.frame(  
 speaker = beszedek\_wf$x@docvars$felszolalo,  
 part = beszedek\_wf$x@docvars$part,  
 theta = beszedek\_wf$theta,  
 theta\_se = beszedek\_wf$se.theta  
) %>%   
 mutate(  
 lower = theta - 1.96 \* theta\_se,  
 upper = theta + 1.96 \* theta\_se  
 )  
  
ggplot(dokumentumok\_wf, aes(theta, reorder(speaker, theta))) +  
 geom\_point() +  
 geom\_errorbarh(aes(xmin = lower, xmax = upper), height = 0) +  
 labs(  
 y = NULL,  
 x = expression(theta)  
 )



A párt metaadattal összehasonlíthatjuk az egy párthoz tartozó frakcióvezetők értékeit a facet\_wrap() használatával. Figyeljünk arra hogy az y tengelyen szabadon

ggplot(dokumentumok\_wf, aes(theta, reorder(speaker, theta))) +  
 geom\_point() +  
 geom\_errorbarh(aes(xmin = lower, xmax = upper), height = 0) +  
 labs(  
 y = NULL,  
 x = expression(theta)  
 ) +  
 facet\_wrap(~part, ncol = 1, scales = "free\_y")



## 9.2 Wordscores

A modell illesztést a wordfish-ez hasonlóan a quanteda.textmodels csomagban található textmodel\_wordscores() függvény végzi. A kiinduló dfm ugyanaz mint amit a fejezet elején elkészítettünk, a beszedek\_dfm.

A referencia pontokat dokumentumváltozóként hozzáadjuk a dfm-hez a refrencia\_pont oszlopot, ami NA értéket kap alapértelmezetten. A kiválasztott referencia dokumentumoknál pedig egyenként hozzáadjuk az értékeket. Erre több megoldás is van, az egyszerűbb út, hogy az egyik és másik végletet a -1; 1 intervallummal jelöljük. Ennek a lehetséges alternatívája, hogy egy külső, már validált forrást használunk. Pártok esetén ilyen lehet a Chapel Hill szakértői kérdőívének a pontszámai, a Manifesto projekt által kódolt jobb-bal (*rile*) dimenzió. A lenti példánál mi maradunk az egyszerűbb bináris kódolásnál. A wordfish eredményt alapul véve a két referencia pont a Gulyás Gergely és Szél Bernadett beszédei lesznek.[[40]](#footnote-175) Ezek a 3. és 10. dokumentumok.

docvars(beszedek\_dfm, "referencia\_pont") <- NA  
docvars(beszedek\_dfm, "referencia\_pont")[3] <- -1  
docvars(beszedek\_dfm, "referencia\_pont")[10] <- 1  
  
docvars(beszedek\_dfm)  
#> id felszolalo part referencia\_pont  
#> 1 20142018\_024\_0002\_0002 Vona Gábor (Jobbik) Jobbik NA  
#> 2 20142018\_055\_0002\_0002 Dr. Schiffer András (LMP) LMP NA  
#> 3 20142018\_064\_0002\_0002 Dr. Szél Bernadett (LMP) LMP -1  
#> 4 20142018\_115\_0002\_0002 Tóbiás József (MSZP) MSZP NA  
#> 5 20142018\_158\_0002\_0002 Schmuck Erzsébet (LMP) LMP NA  
#> 6 20142018\_172\_0002\_0002 Dr. Tóth Bertalan (MSZP) MSZP NA  
#> 7 20142018\_206\_0002\_0002 Volner János (Jobbik) Jobbik NA  
#> 8 20142018\_212\_0002\_0002 Kósa Lajos (Fidesz) Fidesz NA  
#> 9 20142018\_236\_0002\_0002 Harrach Péter (KDNP) KDNP NA  
#> 10 20142018\_249\_0002\_0002 Dr. Gulyás Gergely (Fidesz) Fidesz 1

A lenti wordscore modell specifikáció követi a [Laver, Benoit, and Garry](#ref-laver2003extracting) ([2003](#ref-laver2003extracting)) - ben leírtakat.

beszedek\_ws <- textmodel\_wordscores(  
 x = beszedek\_dfm,  
 y = docvars(beszedek\_dfm, "referencia\_pont"),  
 scale = "linear",  
 smooth = 0  
 )  
  
summary(beszedek\_ws, 10)  
#>   
#> Call:  
#> textmodel\_wordscores.dfm(x = beszedek\_dfm, y = docvars(beszedek\_dfm,   
#> "referencia\_pont"), scale = "linear", smooth = 0)  
#>   
#> Reference Document Statistics:  
#> score total min max mean median  
#> text1 NA 486 0 18 0.2017 0  
#> text2 NA 395 0 12 0.1639 0  
#> text3 -1 439 0 12 0.1822 0  
#> text4 NA 330 0 7 0.1369 0  
#> text5 NA 360 0 8 0.1494 0  
#> text6 NA 328 0 5 0.1361 0  
#> text7 NA 349 0 5 0.1448 0  
#> text8 NA 387 0 10 0.1606 0  
#> text9 NA 307 0 13 0.1274 0  
#> text10 1 383 0 8 0.1589 0  
#>   
#> Wordscores:  
#> (showing first 10 elements)  
#> tisztelt elnök úr országgyulés ország is   
#> -0.07547 0.39255 0.06813 0.06813 -1.00000 -0.19859   
#> sot nemhogy tette fidesz   
#> -1.00000 -1.00000 -1.00000 1.00000

Az illesztett wordscores modellünkkel ezek után már meg tudjuk becsülni a korpuszban lévő többi dokumentum pozícióját. Ehhez az R beépített predict() megoldását használjuk. A kiegészítő opciókkal a konfidencia intervallum alsó és felső határát is meg tudjuk becsülni, ami jól jön hogyha szeretnénk ábrázolni az eredményt.

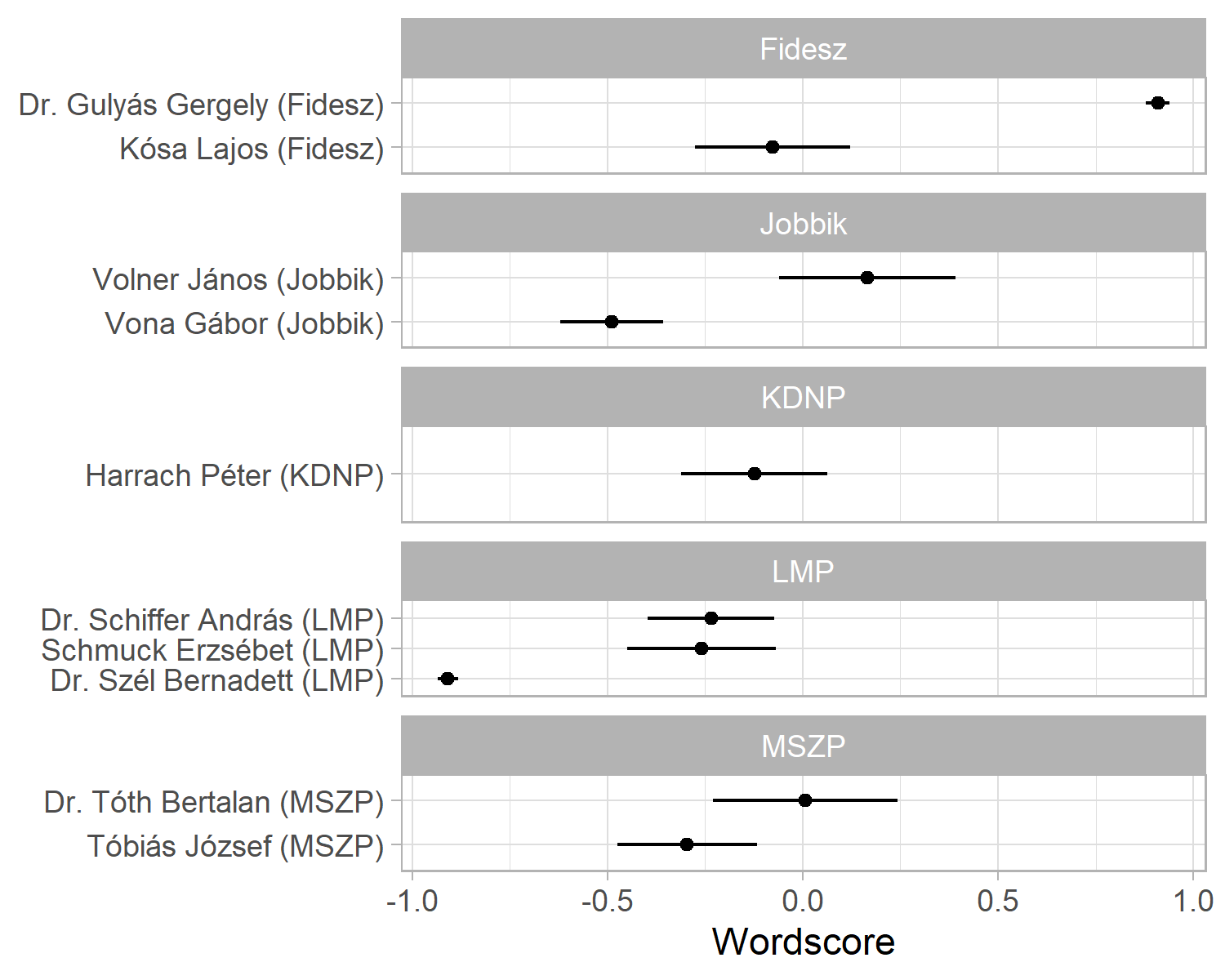
beszedek\_ws\_pred <- predict(  
 beszedek\_ws,   
 newdata = beszedek\_dfm,  
 interval = "confidence")  
  
beszedek\_ws\_pred <- as.data.frame(beszedek\_ws\_pred$fit)  
  
beszedek\_ws\_pred  
#> fit lwr upr  
#> text1 -0.489860579 -0.62138707 -0.35833409  
#> text2 -0.234609623 -0.39658117 -0.07263807  
#> text3 -0.909048451 -0.93507086 -0.88302605  
#> text4 -0.296528588 -0.47539855 -0.11765863  
#> text5 -0.259074418 -0.44948427 -0.06866457  
#> text6 0.006320468 -0.23056645 0.24320738  
#> text7 0.165042014 -0.06144022 0.39152425  
#> text8 -0.077739857 -0.27645536 0.12097565  
#> text9 -0.123985348 -0.31176579 0.06379509  
#> text10 0.909048451 0.87934394 0.93875296

A kapott modellünket a wordfishez hasonlóan tudjuk ábrázolni, miután a beszedek\_ws\_pred objektumból egy data framet csinálunk és a ggplot2-vel elkészítjük a vizualizációt. A dokumentumok\_ws két részből áll össze. Először a wordscores modell objektumunkból a frakcióvezetők neveit és pártjaikat emeljük ki (kicsit körülményes a dolog mert egy komplexebb objektumban tárolja őket a quanteda, de az str() függvény tud segíteni ilyen esetekben). A dokumentumok becsült pontszámait pedig a beszedek\_ws\_pred objektumból készített data frame hozzácsatolásával tesszük meg. Ehhez a dplyr csomag bind\_cols függvényét használjuk. Fontos, hogy itt teljesen biztosnak kell lennünk abban, hogy a sorok a két data frame esetében ugyanarra a dokumentumra vonatkoznak.

dokumentumok\_ws <- data.frame(  
 speaker = beszedek\_ws$x@docvars$felszolalo,  
 part = beszedek\_ws$x@docvars$part  
)  
  
dokumentumok\_ws <- bind\_cols(dokumentumok\_ws, beszedek\_ws\_pred)  
  
dokumentumok\_ws  
#> speaker part fit lwr upr  
#> text1 Vona Gábor (Jobbik) Jobbik -0.489860579 -0.62138707 -0.35833409  
#> text2 Dr. Schiffer András (LMP) LMP -0.234609623 -0.39658117 -0.07263807  
#> text3 Dr. Szél Bernadett (LMP) LMP -0.909048451 -0.93507086 -0.88302605  
#> text4 Tóbiás József (MSZP) MSZP -0.296528588 -0.47539855 -0.11765863  
#> text5 Schmuck Erzsébet (LMP) LMP -0.259074418 -0.44948427 -0.06866457  
#> text6 Dr. Tóth Bertalan (MSZP) MSZP 0.006320468 -0.23056645 0.24320738  
#> text7 Volner János (Jobbik) Jobbik 0.165042014 -0.06144022 0.39152425  
#> text8 Kósa Lajos (Fidesz) Fidesz -0.077739857 -0.27645536 0.12097565  
#> text9 Harrach Péter (KDNP) KDNP -0.123985348 -0.31176579 0.06379509  
#> text10 Dr. Gulyás Gergely (Fidesz) Fidesz 0.909048451 0.87934394 0.93875296

A lenti példánál a párton belüli bontást illusztráljuk, a facet\_wrap() segítségével.

ggplot(dokumentumok\_ws, aes(fit, reorder(speaker, fit))) +  
 geom\_point() +  
 geom\_errorbarh(aes(xmin = lwr, xmax = upr), height = 0) +  
 labs(  
 y = NULL,  
 x = "Wordscore"  
 ) +  
 facet\_wrap(~part, ncol = 1, scales = "free\_y")



# 10 Szövegösszehasonlítás

A gépi szövegösszehasonlítás a mindennapi életünk számos területén megjelenő szövegbányászati technika, bár az emberek többség nincs ennek tudatában. Ezen a módszeren alapulnak a böngészők kereső mechanizmusai, vagy a kérdés-felelet (Q&A) fórumok algoritmusai, melyek ellenőrzik, hogy szerepel-e már a feltenni kívánt kérdés a fórumon ([Sieg 2018](#ref-siegTextSimilaritiesEstimate2018)). Alkalmazzák továbbá a szövegösszehasonlítást a gépi szövegfordításban és az automatikus kérdésmegválaszolási feladatok esetén is ([Wang and Dong 2020](#ref-wangMeasurementTextSimilarity2020)), de akár automatizált esszé értékelésre vagy, plágiumellenőrzésre is hasznosítható az eljárás ([Bar, Zesch, and Gurevych 2011](#ref-barReflectiveViewText2011)).

A szövegösszehasonlítás hétköznapi életben előforduló rejtett alkalmazásain túl a társadalomtudományok művelői is számos esetben hasznosítják az eljárást. A politikatudomány területén többek között használhatjuk arra, hogy eldöntsük, mennyire különböznek egymástól a benyújtott törvényjavaslatok és az elfogadott törvények szövegei, ezzel fontos információhoz jutva arról, hogy milyen szerepe van a parlamenti vitának a végleges törvények kialakításában. Egy másik példa a szakpolitikai prioritásokban és alapelvekben végbemenő változások elemzése, melyet pl. szakpolitikai javaslatok, vagy ilyen témájú viták leiratainak elemzésével is megtehetünk.

A könyv korábbi fejezeteiben bemutatott eljárások között sok olyat találunk, melyek alkalmasak arra, hogy a szövegek hasonlóságából valamilyen információt nyerjünk. Ugyanakkor vannak módszerek, melyek segítségével számszerűsíthetjük a szövegek közötti különbségeket, eza fejezet ezekről nyújt egy rövid áttekintést. Mindenek előtt azonban azt kell tisztáznunk, hogy miként értelmezzük a hasonlóságot. A hasonlóságelemzéseket jellemzően két nagy kategóriába szoktuk sorolni a mérni kívánt hasonlóság típusa szerint. Ez alapján beszélhetünk lexikális (formai) és szemantikai hasonlóságról.

## 10.1 Lexikális hasonlóság

A lexikális hasonlóság a gépi szövegfeldolgozás egy egyszerűbb megközelítése, amikor nem várjuk el az elemzésünktől, hogy a „értse” a szöveget, csupán a formai hasonlóságot figyeljük. A megközelítés előnye, hogy számítási szempontból jelentősen egyszerűbb, mint a szemantikai hasonlóságra irányuló elemzések, hátránya azonban, hogy az egyszerűség könnyen tévútra vihet szofisztikáltabb elemzések esetén, így pl. a lexikális hasonlóság szempontjából az alábbi két példamondat azonosnak tekinthető, hiszen formailag (kifejezések szintjén) megegyeznek.

1. *„A boszorkány megsüti Jancsit és Juliskát”*
2. *„Jancsi és Juliska megsüti a boszorkányt”*

Két dokumentum közötti lexikális hasonlóságot a szöveg számos szintjén mérhetjük: karakterláncok (sztringek), szóalakok (tokenek), n-grammok (n egységből álló karakterláncok), szózsákok (bag of words) között, de akár a dokumentum nagyobb egységei, így szövegrészletek és dokumentumok között is. Bevett megközelítés továbbá a szókészlet összehasonlítása, melyet lexikális és szemantikai hasonlóság feltárására egyaránt használhatunk.

A hasonlóság számítására számos metrika létezik (lásd: [Wang and Dong](#ref-wangMeasurementTextSimilarity2020) ([2020](#ref-wangMeasurementTextSimilarity2020))). Ezek jelentős része valamilyen távolságszámításon alapul, mint pl. a koszinusz vagy manhattan távolságon alapuló szöveghasonlóság. A koszinusz távolság a két szövegvektor (azaz szöveg vektorizált formája) által bezárt szögben határozza meg a dokumentumok távolságát ([Wang and Dong 2020](#ref-wangMeasurementTextSimilarity2020)), míg a manhattan távolság a horizontális és vertikális távolságok összegeként számítja azt. ([Ladd 2020](#ref-laddUnderstandingUsingCommon2020)). Széles körben alkalmazott dokumentumhasonlósági metrika továbbá a Jaccard hasonlóság, melynek számítása egy egyszerű eljáráson alapul: a két dokumentumban egyező szavak számát elosztja a két dokumentumban szereplő szavak számának uniójának (vagyis az két dokumentum szavainak számának összegével, melyből kivonja az egyező szavak számának összegét). A jaccard hasonlóság tehát azt képes megmutatni, hogy a két dokumentum teljes szószámához képest mekkora az azonos kifejezések aránya.

$$$$

([Wang and Dong 2020, 6](#ref-wangMeasurementTextSimilarity2020).)

## 10.2 Szemantikai hasonlóság

A szemantikai hasonlóság a lexikai hasonlósággal szemben egy komplexebb számítást, melynek során az algoritmus a szavak tartalmát is képes a elemezni tenni. Így például formai szempontból hiába nem azonos az alábbi két példamondat, a szemantikai hasonlóságvizsgálatnak észlelnie kell a tartalmi azonosságot.

1. *A diákok jegyzetelnek amíg a professzor előadást tart.*
2. *A nebulók írnak, amikor az oktató beszél.*

A jelentésbeli hasonlóság kimutatására számos megközelítés létezik. Többek között alkalmazható a témamodellezés (topicmodellezés) – melyet a Felügyelet nélküli tanulás: Topik modellezés magyar törvényszövegeken című fejezetben tárgyalunk bővebben **KERESZTHIVATKOZÁS A TM FEJEZETRE**– ezen belül pedig az LDA látens dirillecht allokáció (*Latent Dirillecht Allocation*,) valamint az LSA (látens érzelem elemzés (*Latent Sentiment Analysis*) is nagyszerű lehetőséget kínálnak arra, hogy az egyes dokumentumainkat tartalmi hasonlóságok alapján csoportosítsuk.

Az LSA-nél és LDA-nél azonban egy fokkal komplexebb megközelítés a szóbeágyazás, melyet a Szóbeágyazások című fejezetben mutatunk be **KERESZTHIVATKOZÁS**. Ez a módszertan a témamodellezéshez képest a szöveg mélyebb szemantikai tartalmait is képes feltárni, hiszen a beágyazásnak köszönhetően képes formailag különböző, de jelentésükben azonos kifejezések azonosságát megmutatni. A jelentésbeli hasonlóság megállapítható a beágyazás során létrehozott vektorreprezentációkból (emlékezzünk: a hasonló vektorreprezentáció hasonló szemantikai tartalomra utal). Kimutathatjuk a szemantikai közelséget például a király – férfi – lovag kifejezések között, de olyan mesterségesen létrehozott jelentésbeli azonosságokat is feltárhatunk, mint az irányítószámok és az általuk jelölt városnevek kapcsolata. Abban az esetben, ha a szóbeágyazást kimondottan a szöveghasonlóság megállapítására szeretnénk használni, a Word Mover’s Distance (WMD) metrikát érdemes használni, mely a vektortérben elhelyezkedő szóvektorok közötti távolság által számszerűsíti a szövegek hasonlóságát([**kusnerWordEmbeddingsDocument2015setwd?**](#Xc4acfb646fc47ec86addb1a30974b2f159d65e4)).

## 10.3 Hasonlóság számítás a gyakorlatban

### 10.3.1 Adatbázis importálás és előkészítés

A fejezet második felében a lexikai hasonlóságvizsgálatra, ezen belül a Jaccard hasonlóság és a Koszinusz hasonlóság számítására mutatunk be egy-egy példát a törvényjavaslatok és az elfogadott törvények szövegeinek összehasonlításával. Az alábbiakban bemuatatott elemzés a **Viscosity Revisited: The Power of Legislatures in New and Old Democracies - A Comparative Text Reuse Analysis (under publication)** című cikkből meríti elemzési főkuszát. Az eredeti cikk által megvalósított elemzést a svájci korpusz elemzése nélkül, a magyar korpusz egy részhalmazán replikáljuk az alábbiakban. A kutatási kérdés arra irányul, hogy mennyiben változik meg a törvényjavaslatok szövege a parlamenti vita folyamán, amíg a javaslat elfogadásra kerül. Az elemzés során a különböző kormányzati ciklusok közötti eltérésekre világítunk rá. Az elemzés megkezdése előtt a már ismert módon betöltjük a szükséges csomagokat: readr, stringr, dplyr, quanteda, readtext, ggplot2.

library(readr)  
library(stringr)  
library(dplyr)  
library(quanteda)  
library(readtext)  
library(ggplot2)

Ezt követően betöltjük azokat az adatbázisokat, amelyeken a szövegösszehasonlítást fogjuk végezni: az elfogadott törvények szövegét tartalmazó korpuszt, a törvényjavaslatok szövegét tartalmazó korpuszt, valamint az ezek összekapcsolását segítő adatbázist, melyben az összetartozó törvényjavaslatok és törvények azonosítóját (id-ját) tároltuk el. Ahogy behívjuk a három CSV-t, érdemes rögtön lekérni az oszlopneveket colnames() és a táblázat dimenzióit dim(), hogy lássuk milyen adatok állnak a rendelkezésünkre, és mekkorák a táblákkal fogunk dolgozni. A dim() függvény első értéke a sorok száma a második pedig az oszlopok száma lesz az adott táblázatban.

torvenyek <- read\_csv("data/torveny\_korpusz.csv")  
colnames(torvenyek)  
dim(torvenyek)

tv\_javaslatok <- read.csv("data/tv\_javaslat\_korpusz.csv")  
colnames(tv\_javaslatok)  
dim(tv\_javaslatok)

parok <- read.csv("data/parok.csv")  
colnames(parok)  
dim(parok)

A beimportált adatbázisok megfigyeléseinek száma egységesen 600. Ez a több mint háromezer megfigyelést tartalmazó eredeti korpusz egy részhalmaza, mely gyorsabb és egyszerűbb elemzést tesz lehetővé. Az oszlopnevek lekérésével láthatjuk, hogy a törvény korpuszban van néhány metaadat, amelyet az elemzés során felhasználhatunk, ezek a kormányzati ciklusra, törvény elfogadásának évére, valamint a benyújtó kormányzati vagy ellenzéki pártállására vonatkoznak. Ezen kívül rendelkezésre állnak a törvényeket és törvényjavaslatokat azonosító kódok (tv\_id és tvjav\_id) melyek segítségével majd tudjuk párosítani az összetartozó törvényjavaslatok és törvények szövegeit. Ezt a left\_join() függvénnyel teszünk meg. Elsőként a törvényeket tartalmazó adatbázishoz kapcsoljuk hozzá a törvény–törvényjavaslat párokat tartalmazó adatbázist a törvények azonosítója (tv\_id) alapján. A colnames() függvény használatával ellenőrizhetjük, hogy sikeres volt-e a művelet, és az új táblában szerepelnek-e a kívánt oszlopok.

tv\_tvjavid\_osszekapcs <- left\_join(torvenyek, parok, by = "tv\_id")  
colnames(tv\_tvjavid\_osszekapcs)  
dim(tv\_tvjavid\_osszekapcs)

Második lépésben a törvényjavaslatokat tartalmazó adatbázist rendeljük hozzá az előzőekben már összekapcsolt két adatbázishoz.

tv\_tvjav\_minta <- left\_join(tv\_tvjavid\_osszekapcs, tv\_javaslatok, by = "tvjav\_id")  
colnames(tv\_tvjav\_minta)  
dim(tv\_tvjav\_minta)

Ha jól végeztük a dolgunkat az adatbázisok összekapcsolása során, az eljárás végére 7 oszlopunk és 600 sorunk van, vagyis az újonnan létrehozott adatbázisba bekerült az összes változó (oszlop). A korpuszaink egy adattáblában való kezelése azért hasznos, mert így nem kell párhuzamosan elvágezni az azonos műveleteket a két korpusz, a törvények és a törvényjavaslatok tisztításához, hanem párhuzamosan tudunk dolgozni a kettővel. Kicsit közelebbről megvizsgálva az adatbázist azt láthatjuk, hogy minden adatbázisunkban szereplő kormányzati ciklusra 100 megfigyelés áll rendelkezésünkre tv\_tvjav\_minta %>% count(korm\_ciklus). Hasonlóan ellenőrizhetjük az egyes évekre eső megfigyelések számát: tv\_tvjav\_minta %>% count(év).

tv\_tvjav\_minta %>%   
 count(korm\_ciklus)

## 10.4 Szövegtisztítás

Mivel az elemzés során két különböző korpusszal dolgozunk – két oszlopnyi szöveggel – egyszerűbb, ha a szövegtisztítás lépéseiből létrehozunk egy külön függvényt, amely magában foglalja a művelet egyes lépéseit, és lehetővé teszi, hogy ne kelljen minden szövegtisztítási lépést külön definiálni az egyes korpuszok esetén.

A függvény neve jelen esetben szovegtisztitas lesz, és a már ismert lépéseket foglalja magában: kontrol karakterek szóközzé alakítása, központozás és a számok eltávolítása. Kisbetűsítés, ismétlődő sztringek és a sztringek előtt található szóközök eltávolítása. Továbbá a str\_remove\_all() függvénnyel eltávolítjuk azokat az írásjeleket, amelyek előfordulnak a szövegben, de számunkra nem hasznosak.

A függvény definiálását az alábbi szintaxissal tehetjük meg.

# fuggveny <- function(bemenet) {  
# elvegzendo\_lepesek  
# return(kimenet)  
# }

A *bemenet* helyen azt jelöljük, hogy milyen objektumon fogjuk végrehajtani a műveleteket, a kimenetet pedig a return() függvénnyel definiáljuk, ez lesz a függvényünk úgynevezett visszatérési értéke, vagyis a *elvégzendő lepesek* szerint átalakított objektum. A szövegtisztító függvény bemeneti és kimeneti értéke is text lesz, mivel ebbe a változóba mentettük az elvégzendő változtatásokat.

szovegtisztitas <- function(text) {  
 text = str\_replace(text, "[:cntrl:]", " ")  
 text = str\_remove\_all(string = text, pattern = "[:punct:]")  
 text = str\_remove\_all(string = text, pattern = "[:digit:]")  
 text = str\_to\_lower(text)  
 text = str\_trim(text)  
 text = str\_squish(text)  
 text = str\_remove\_all(string = text, pattern = "’")  
 text = str\_remove\_all(string = text, pattern = "…")  
 text = str\_remove\_all(string = text, pattern = "–")  
 text = str\_remove\_all(string = text, pattern = "“")  
 text = str\_remove\_all(string = text, pattern = "”")  
 text = str\_remove\_all(string = text, pattern = "„")  
 text = str\_remove\_all(string = text, pattern = "«")  
 text = str\_remove\_all(string = text, pattern = "»")  
 text = str\_remove\_all(string = text, pattern = "§")  
 text = str\_remove\_all(string = text, pattern = "°")  
 text = str\_remove\_all(string = text, pattern = "<U+25A1>")  
 text = str\_remove\_all(string = text, pattern = "@")  
 return(text)  
}

Miután létrehoztuk a szövegtisztításra alkalmas függvényünket, az adatbázis két oszlopára fogjuk alkalmazni: a törvények szövegét és törvényjavaslatok szövegét tartalmazó oszlopra, melyben a mapply() függvény lesz a segítségünkre. A mapply() függvényen belül megadjuk az adatbázist, és ennek vonatkozó részeire való hivatkozást tv\_tvjav\_minta[ ,c("torveny\_szoveg","tvjav\_szoveg")]. Az alkalmazni kívánt függvényt a FUN argumanetumaként adhatjuk meg – értelemszerűen, ez esetünkben az előzőekben létrehozott szovegtisztitas függvény lesz. Végezetül pedig a fügvényünk által megtisztított új oszlopokkal felülírjuk az előző adatbázisunk vonatkozó oszlopait, vagyis a torveny\_szoveg és a tvjav\_szoveg oszlopokat: tv\_tvjav\_minta[, c("torveny\_szoveg","tvjav\_szoveg")] <- >>újonnan létrehozott oszlopok<<.

tv\_tvjav\_minta[, c("torveny\_szoveg","tvjav\_szoveg")] <- mapply(tv\_tvjav\_minta[ ,c("torveny\_szoveg","tvjav\_szoveg")], FUN = szovegtisztitas)

A szövegtisztítás következő lépése a stopszavak meghatározása és kiszűrése a szövegből. Itt a quanteda csomagban elérhető magyar nyelvű stopszavakat, valamint a 9. fejezetben meghatározott speciális jogi stopszavak listáját használjuk.

legal\_stopwords <- readtext("data/legal\_stopwords.csv", encoding = "UTF8") %>%  
 pull(text)

A stopszavak beimportálását követően korpusszá alakítjuk a szövegeinket és tokenizáljuk azokat. Ezt már külön-külön végezzük el a törvények és törvényjavaslatok szövegeire, azonos lépésekben haladva.A létrehozott objektumokat itt is ellenőrizhetjük pl. a summary(torvenyek\_coprus) paranccsal, vagy a torvenyek\_tokens[1:3] paranccsal, mely az első 3 dokumentum tokenjeit fogja megmutatni.

torvenyek\_corpus <- corpus(tv\_tvjav\_minta$torveny\_szoveg)  
tv\_javaslatok\_corpus <- corpus(tv\_tvjav\_minta$tvjav\_szoveg)

torvenyek\_tokens <- tokens(torvenyek\_corpus) %>%  
 tokens\_remove(stopwords("hungarian")) %>%  
 tokens\_remove(legal\_stopwords) %>%  
 tokens\_wordstem(language = "hun")  
  
tv\_javaslatok\_tokens <- tokens(tv\_javaslatok\_corpus) %>%  
 tokens\_remove(stopwords("hungarian")) %>%  
 tokens\_remove(legal\_stopwords) %>%  
 tokens\_wordstem(language = "hun")

A szövegek tokenizálásával és a stopszavak eltávolításával a szövegtisztítás végére értünk, így megkezdhetjük az elemzést.

## 10.5 Jaccard hasonlóság számítás

A Jaccard hasonlóság kiszámításához a quanteda textstat\_simil() függvényét fogjuk alkalmazni. Mivel a textstat\_simil() függvény dokumentum-kifejezés mátrixot vár bemenetként, elsőként alakítsuk át ennek megfelelően a korpuszainkat.

torvenyek\_dfm <- dfm(torvenyek\_tokens)  
tv\_javaslatok\_dfm <- dfm(tv\_javaslatok\_tokens)

Miután létrehoztuk a dokumentum-kifejezés mátrixokat, érdemes a leggyakoribb tokeneket ellenőrizni a textstat\_frequency() függvénnyel, hogy biztosak lehssünk benne, hogy a megfelelő eredményt értük el a szövegtisztítás során. (Amennyiben nem vagyunk elégedettek, érdemes visszatérni a stopszavakhoz és újabb kifejezezéseket hozzárendelni a stopszó listához).

tv\_toptokens <- textstat\_frequency(torvenyek\_dfm, n = 15)  
tv\_toptokens

tvjav\_toptokens <- textstat\_frequency(tv\_javaslatok\_dfm, n = 15)  
tvjav\_toptokens

A létrehozott dokumentum-kifejezés mátrixokon elvégezhetjük a dokumentumhasonlóság vizsgálatot. (A jaccard hasonlóság metrika, illetve quanteda textstat\_simil() függvénye alkalmazható egy korpuszra is. Egy korpuszra végezve az elemzést a korpusz dokumentumai közötti hasonlóságot számítja ki a függvény, míg két korpuszra mindkét korpusz összes dokumentuma közötti hasonlóságot. Érdemes továbbá azt is megjegyezni, hogy a textstat\_simil() method argumentumaként megadható számos más hasonlósági metrika is, pl. a koszinusz hasonlóság (method = "cosine"), melyekkel számos további érdekes számítás végezhető. Bővebben a textstat\_simil() függény használatáról és argumentumairól a quanteda hivatalos honlapján olvashatunk: [textstat\_simil( )](https://quanteda.io/reference/textstat_simil.html)). A textstat\_simil() függvény kapcsán azt is érdemes figyelembe venni, hogy mivel nem csak a dokumentum párokra, hanem az összes bemenetként megadott dokumentumra külön kiszámítja a jaccard indexet a korpusz(ok) méretének növelésével a számítás kapacítás- és időigényessége exponenciálisan növekszik. Két 600 dokumentumból álló korpusz esetén kb. 4-5 perc a számítási idő, míg 360 dokumentum estén csupán 1-2 perc.

jaccard\_hasonlosag = textstat\_simil(torvenyek\_dfm, tv\_javaslatok\_dfm, method = "jaccard")

Mivel az eredménymátrixunk meglehetősen terjedelmes, nem érdemes az egészet egyben megtekinteni, egyszerűbb az első néhány dokumentum közötti hasonlóságra szűrni, melyet a szögletes zárójelben való indexeléssel tudunk megtenni. Az [1:5, 1:5] kifejezéssel specifikálhatjuk az elsőtől az ötödikig a sorokat és az oszlopkat.

jaccard\_hasonlosag[1:5, 1:5]

A mátrix fődiagonáljában jelennek meg az összetartozó törvényekre és törvényszövegek vonatkozó értékek, minden más érték nem összetartozó törvény és törvényjavaslat szövegek hasonlóságára vonatkozik, vagyis a vizsgálatunk szempontjából irreleváns. Ahhoz, hogy kinyerjük a számunkra értékes adatokat a jaccard hasonlóság változót mátrixszá kell alakítani. (Ránézésre úgy tűnhet, hogy már most is mátrix, de valójában ez egy speciális S4 típusú objektum, melyben a mátrixon kívül más típusú információk is elvannak mentve. Az egyes objektumok típusát mindig ellenőrizhetjük a typeof() függvénnyel: typeof(jaccard\_hasonlosag)). A mátrixszá alakításban az as.matrix() függvény lesz a segítségünkre, melynek egyúttal a diagonálját is kinyerhetjük a diag() függvénnyel. Ha jól dolgoztunk a létrehozott jaccard\_diag első öt eleme (jaccard\_diag[1:5]) megegyezik a fent megjelenített 5x5-ös mátrix fődiagonáljában elhelyezkedő értékekkel, hossza pedig (length()) a mátrix bármelyik dimenziójával.

jaccard\_diag <- diag(as.matrix(jaccard\_hasonlosag))  
jaccard\_diag[1:5]

Miután siekrült kinyerni az egyes törvny–törvényjavaslat párokra vonatkozó jaccard értéket, érdemes a számításainakt hozzárendelni az eredeti adattáblánkhoz, hogy a meglévő metaadatok fényében tudjuk kiértékelni az egyes dokumentumok közötti hasonlóságot. A hozzárendeléshez egyszerűen definiálunk egy új oszlopot a meglévő adatbázisban tv\_tvjav\_minta$jaccard\_index, melyhez hozzárendeljük a diagonálból kinyert értékeket.

tv\_tvjav\_minta$jaccard\_index <- jaccard\_diag

Érdemes megnézni a végeredményt, ellenőrizni a jaccard hasonlóság legmagasabb és legalacsonyabb értékeit. A top\_n() függvény használatával ki tudjuk válogatni a legmagasabb és legalacsonyabb értékeket. A top\_n() függvény első argumentuma a változó lesz, ami alapján a legalacsonyabb és legmagasabb értékeket keressük, a második argumentum pedig azt specifikálja, hogy a legmagasabb és legalacsonyabb értékek közül hányat szeretnénk látni. Az n=5 értékkel a legmagasabb, az n=-5 értékkel a legalacsonyabb 5 jaccard indexszel rendelkező sort tudjuk kiszűrni. Emellett érdemes arra is odafigyelni, hogy a szövegeket tartalmazó oszlopainkat ne próbáljuk meg kiíratni, hiszen ez jelentősen lelassítja az RStudio működését és csökkenti a kiírt eredmények áttekinthetőségét.

tv\_tvjav\_minta[, c("tv\_id", "korm\_ciklus", "tvjav\_id", "jaccard\_index")] %>%  
 top\_n(jaccard\_index, n = 5)

tv\_tvjav\_minta[, c("tv\_id", "korm\_ciklus", "tvjav\_id", "jaccard\_index")] %>%   
 top\_n(jaccard\_index, n=-5)

Láthatjuk, hogy az 5 öt leghasonlóbb törvény–törvényjavaslat pár esetén 0.98 felett van a jaccard hasonlág értéke, míg a leginkább különböző 5-nél 0.03 alatt.

## 10.6 Koszinusz hasonlóság számítás

A Jaccard hasonlóság számítás után a koszinusz távolság számítása már nem jelent nagy kihívást, hiszen a textat\_simil() függvénnyel ezt is kiszámíthatjuk, csupán a metrika paramétereként (method =) megadhatjuk a kosziniszt is. Ahogy az előbbiekben, itt is a dokumentum-kifejezés mátrixokat adjuk meg bemeneti értékként.

koszinusz\_hasonlosag <- textstat\_simil(torvenyek\_dfm, tv\_javaslatok\_dfm, method = "cosine")

Érdmes itt is megtekinteni a mátrix első néhány sorába és oszlopába eső értékeket.

koszinusz\_hasonlosag[0:5, 0:5]

Ebben az esetben is a mátrix diagonálja van csak szükségünk, melyet a fent ismeretett módon nyerünk ki a mátrixból.

koszinusz\_diag <- diag(as.matrix(koszinusz\_hasonlosag))  
koszinusz\_diag[1:5]

Végezetül pedig a koszinusz értékeket is hozzárendeljük egy

tv\_tvjav\_minta$koszinusz <- koszinusz\_diag

colnames(tv\_tvjav\_minta)

## 10.7 Az eredmények vizualizációja

A hasonlóság metrikák vizulizációjára gyakran alkalmazott megoldás a hőtérkép ( **heatmap**) mellyel korrelációs mátrixokat ábrázolhatunk. Ebben az esetben a mátrix értékeit egy színskálán vizualizáljuk, ahol a világosabb színek a magasabb, a sötétebb színek az alacsonyabb értékeket jelölik. A jaccard hasonlóság számítás és a koszinusz hasonlóság számításakor kapott mátrixok esetén is ábrázoljhatjuk az értékeinket ilyen módon. Mivel azonban mindkét mátrix 600x600-as, nem érdemes a teljes mátrixot megjelentíeni, mert ilyen nagy mennyiségű adatnál már értelmezhetetlenné válik az ábra, így csak az utlsó 100 elemet, vagyis a 2014-2018-as időszakra vonatkozó értékeket jelenítjük meg. Ezt a kosziunsz\_hasonlóság nevű objektumunk feldarabolásával tesszük meg, szögletes zárójelben jelölve, hogy a mátrix mely sorait, és mely oszlopait szeretnénk használni: ‘koszinusz\_hasonlosag[501:600, 501:600].’ Mivel az R még nem mátrixként értelmezi ezt az objektumot (lásd: typeof()) át kell alakítanunk mátrixszá az as.matrix() függvény segítségével. Ezt követően a matrix\_raster\_plot() függvényt használjuk az ábra vizualizációjához.[[41]](#footnote-187)

library(nandb)  
heat\_koszinusz <- matrix\_raster\_plot(as.matrix(koszinusz\_hasonlosag[501:600, 501:600]), scale\_name = "Koszinusz hasonlóság")  
heat\_koszinusz

heat\_jaccard <- matrix\_raster\_plot(as.matrix(jaccard\_hasonlosag[501:600, 501:600]), scale\_name = "Jaccard hasonlóság")  
heat\_jaccard

A két plot összehasonlításanál láthatjuk, hogy a koszinusz hasonlóság általában magasabb hasonlósági értékeket mutat. A mátrix főátlójában kiugró világos csík azt mutatja meg, hogy a legnagyobb hasonlóság az összetartozó törvény-törvényjavaslat szövegek között mutatkozik meg, eddig tehát az adataink az elvárásaink szerinti képet mutatják – amennyiben a világos csíkot nem látnánk, az egyértelmű visszajelzés volna arról, hogy elrontottunk valamit a szövegelőkészítés eddigi lépéseinek folyán, vagy a várakozásásaink voltak teljesen rosszak.

A koszinusz és a jaccard hasonlóság értékét ábrázolhatjuk közös pontdiagrammon a geom\_jitter() segítségével.

ggplot(tv\_tvjav\_minta, aes(x=év))+   
geom\_jitter(aes(y = jaccard\_index), color = "navyblue", size = 1.2) +  
geom\_jitter(aes(y = koszinusz), color = "deeppink3", size = 1.2)+  
ylab('Jaccard és koszinusz hasonóság')+  
xlab('Év')  
### Ide még kellene egy legend.

A hasonlósági értékek évenkétni alakulásának megértése érdekében érdemes átlagot számolni a mutatókra. Ezt a group\_by() és a summarize() függvények együttes alkalmazásával tehetjük meg. Megadjuk, hogy évenkénti bontásban szeretnénk a számításainkat elvégezni group\_by(év), és azt, hogy átlag számítást szeretnénk végezni mean().

evenkenti\_atlag <- tv\_tvjav\_minta %>%   
 group\_by(év) %>%   
 summarize(átl\_koszinusz = mean(koszinusz), átl\_jacc = mean(jaccard\_index))  
  
evenkenti\_atlag

Az évenkénti átlagot tartalmazó adattáblánkara ezt követően vonal diagramot illesztünk.

ggplot(data=evenkenti\_atlag, aes(x=év)) +  
 geom\_line(aes(y=átl\_koszinusz), color = 'navyblue')+  
 geom\_line(aes(y=átl\_jacc), color= 'deeppink3')+  
 ylab('Jaccard és koszinusz hasonóság évenkénti átlagérték')+  
 xlab('Év')  
### Ide még kellene egy legend.

Ahhoz, hogy valamivel pontosabb képet kapjunk a jaccard index értékének alakulásáról, érdemes vizualizálni őket a ggplot segítségével. Elsőként évenkénti bontásban ábrázoljuk a jaccard index értékének alakulását. A színek kormányzati ciklusok szerinti bontását a color = komr\_ciklus változóval tudjuk megadni. A tengelyfeliratokat az xlab() és ylab() paramétereiként adjuk meg, a jelmagyarázat címét pedig a labs() értékeként. A ggplot magától csak néhány értéket rendelne az x tengelyhez feliratként, amit átállíthatunk a scale\_x\_continuous() fügvénnyel és ezek breaks és labels paramétereivel – előbbi az adatfeliratok helyét, utóbbi az adatfeliratok szövegét specifikálja. Értékeiket megadhatjuk manuálisan is egy karaktervektorként, még egyszerűbben pedig úgy specifikálhatjuk, hogy az év oszlopunk egyedi értékeire hivatkozunk: unique(tv\_tvjav\_minta$év). Végezetül pedig a beállítjuk, hogy az x tengelyen legyenek elforgatva a tengelyfeliratok, hogy ne takarják ki egymást, a forgatás mértékét itt szögben adhatjuk meg: angle = 45.

ggplot(data = tv\_tvjav\_minta, aes((év), jaccard\_index, color = korm\_ciklus)) +  
 geom\_point() +  
 xlab("Év") +  
 ylab("Jaccard hasonlóság")+  
 labs(color = "Kormányzati ciklus")+  
 scale\_x\_continuous(breaks=c(unique(tv\_tvjav\_minta$év)), labels=c(unique(tv\_tvjav\_minta$év)))+  
 theme(axis.text.x = element\_text(angle = 45))

A pontdiagram látványos, de esetünkben kevés érdemi információ derül ki róla. A második ábránkon boxplotokkal fogjuk ábrázolni a jaccard hasonlóság alakulását.

ggplot(tv\_tvjav\_minta, aes(factor(év), jaccard\_index, color = korm\_ciklus)) +  
 geom\_boxplot() +  
 xlab("Év") +  
 ylab("Jaccard hasonlóság")+   
 labs(color = "Kormányzati ciklus")+  
 theme(axis.text.x = element\_text(angle = 45))

Az ábrán szembetűnő a 2003-as év kiugróan alacsony értéke, azonban itt érdemes figyelembe venni - ami a pontdiagramról is leolvasható - hogy 2003-ra csupán 2 adatpont áll rendelkezésre. A legalacsonyabb jaccard hasonlóság talán az 1994-1998-as időszakra jellemző, míg a 2014-2018-as iőszakra szembetűnően magas jaccard értékeket látunk az első ábrázolt ciklushoz képest. Összességében nehéz trendet látni az ábrán, de érdemes azt is megjegyezni, hogy a negatív irányba kiugró adatpontok a 2014-2018-as ciklusban jelentősen nagyobb arányban tűnnek fel, mint a korábbi kormányzati ciklusok alatt.

Végezetül pedig ábrázolhatjuk a jaccard hasonlóságot a benyújtó személye alapján is a korm\_ell változónk alapján. A változók értékei a következőek [CAP kódkönyve alapján](https://docs.google.com/document/d/11AwjQiRNbMifBaBnbgo2MpWR00-03VEjb63FgPUy6VA/edit#heading=h.cx672ghhctgs) (p.4):

*0 - Ellenzéki benyújtó*

*1 - Kormánypárti benyújtó*

*2 - Kormánypárti és ellenzéki benyújtó közösen*

*3 - Benyújtók legalább két ellenzéki pártból*

*4 - Benyújtók legalább két ellenzéki pártból*

*5 - Nem releváns - a benyújtó a kabinet tagja*

*6 - Nem releváns - a benyújtó a bizottság tagja volt*

*7 - Nem releváns - a benyújtó sem a parlamentnek, sem a kabinetnek, sem a bizottságnak nem tagja*

######### EZT TÖRÖLNI!!!!!!!!!!!! - ha működik a tidybb megoldás#############  
  
# Mivel nincs túl sok adatpontunk, és ezek többsége a 900-as értéket vesz fel (lásd: `tv\_tvjav\_minta %>% count(korm\_ell)`), érdemes összevonni a 0-ás és a 3-as változót, valamint az 1-es és a 4-es változót egy-egy értékbe, hogy jobban elemezhetőek legyenek az eredményeink. Ehhez a `korm\_ell` változó értékei alapján definiálunk egy új `korm\_ell2` változót. Az új változó definiálását és az értékadásokat a már ismert szögletes zárójellel, az adatbázis bizonyos részeire hivatkozva fogjuk megtenni. Először csupa `NA` értékkel létrehozzuk azt az oszlopot, amibe szeretetnénk az új értékeket elmenteni. A `tv\_tvjav\_minta$korm\_ell2 <- 0` parancs a `korm\_ell2` változó minden értékéhez 0-t rendelne. Ahhoz, hogy csak a `korm\_ell` oszlopban 0-ás, vagy 3-as értéket felvevő sorokban adjunk 0-ás értéket, a következő módon szűrjük az adatbázist: `[tv\_tvjav\_minta$korm\_ell == 0 | tv\_tvjav\_minta$korm\_ell == 3]`. A `|` jel a "vagy"-ot jelöl. Ugyanezzel a módszerrel a többi értéket is hozzárendeljük a `korm\_ell2` változóhoz.  
  
  
tv\_tvjav\_minta$korm\_ell2 <- NA  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 0 | tv\_tvjav\_minta$korm\_ell == 3] <- 0  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 1 | tv\_tvjav\_minta$korm\_ell == 4] <- 1  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 2] <- 2  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 900] <- 900  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 901] <- 901  
tv\_tvjav\_minta$korm\_ell2[tv\_tvjav\_minta$korm\_ell == 902] <- 902

Mivel nincs túl sok adatpontunk, és ezek többsége a 900-as adatpont alá esik (lásd: tv\_tvjav\_minta %>% count(korm\_ell)), érdemes összevonni a 0-ás és a 3-as változót, valamint az 1-es és a 4-es változót egy-egy értékbe, hogy jobban elemezhetőek legyenek az eredményeink. Ehhez a korm\_ell változó értékei alapján definiálunk egy új korm\_ell2 változót. Az új változó definiálását és az értékadásokat a dplyr case\_when() függvényével fogjuk megtenni. A függvényen belül a baloldalra kerül, hogy milyen értékek alapján szeretnénk az új értéket meghatározni a tilde (~) után pedig az, hogy mi legyen az újonnan létrehozott oszlop értéke. Tehát a case\_when()-en belül lévő első sor azt fejezi ki, hogy amennyiben a korm\_ell egyenlő 0-val, vagy (|) a korm\_ell egyenlő 3-mal, legyen a korm\_ell2 értéke 0.

# tv\_tvjav\_minta <- tv\_tvjav\_minta %>%   
# mutate(korm\_ell2 = case\_when(korm\_ell == 0 | korm\_ell == 3 ~ 0,   
# korm\_ell == 1 | korm\_ell == 4 ~ 1,  
# korm\_ell == 2 ~ 2,   
# korm\_ell == 900 ~ 900,  
# korm\_ell == 901 ~ 901,   
# korm\_ell == 902 ~ 902))

Miután létrehoztuk az új oszlopot, létrehozhatjuk a vizualizációt is annak alapján. Itt egy speciális pontdiagramot fogunk használni geom\_jitter()-t,ami annyiban különbözik a pontdiagramtól, hogy kicsit szórtabban ábrázolja a diszkrét értékekre (évek) eső pontokat, hogy az egy helyen sűrűsödő értékek ne takarják ki egymást.

A pontok színeit manuálisan is definiálhatjuk, ha szeretnénk kontrollálni, hogy melyik érték milyen színben tűnjön fel. Ezt a scale\_color\_manual() függvény hozzárendelésével tehetjük meg. Ebben az esetben a breaks, a values és a labels paramétereket adjuk meg. Mivel a labels() értékei ebben az esetben hosszabb feliratok, érdemes külön objektumba (korm\_ell\_labels) elmenteni őket, hogy az ábra kódjában már csak az objektum nevére kelljen hivatkozni, ezzel átláthatóbbá téve a kódot által. A values változóban definiálhatjuk a felhasználni kívánt színeket, amit akár hexadecimális RGB kódok, akár a színek nevének megadásával is megtehetünk. Emellett az R számos előre definiált palettátt is felkínál, melyek alapján automatikusan hozzárendeli a színeket az értékekhez. Bővebben a színekről: [Colors ggplot2](http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/).

Ahhoz, hogy a színeket hozzárendelhessük az értékeinkhez a korm\_ell2 változót faktorrá, vagyis kategorikus értékeket tartalmazó változóvá kell alakítanunk: color = as.factor(korm\_ell2). [Itt picit bizonytalan vagyok, hogy tényleg ez okozz-e a hibát, ami miatt factorrá kell alakítani.] Az előzőekhez képest módosítjuk a jelmagyarázat poziícióját és elrendezését is. A legend.position-t “bottom” értékre állítva az ábra aljára helyezhetjük azt, valamint azt is megadhatjuk, hogy hány oszlopba legyenek tagolva a jelmagyarázat értékei a guides(color=guide\_legend(ncol=3)) sor segítségével.

korm\_ell\_labels = c("Ellenzéki képviselő(k)",  
 "Kormánypárti képviselő(k)",  
 "Kormányzati és ellenzéki képviselők közösen",  
 "Kabinet tagja",   
 "Bizottság tagja",   
 "Egyik sem")  
  
ggplot(tv\_tvjav\_minta, aes(év, jaccard\_index, color = as.factor(korm\_ell2))) +  
 geom\_jitter(size =1.2)+  
 scale\_color\_manual(  
 breaks = c("0", "1", "2", "900", "901", "902"),   
 values = c("slateblue3", "deeppink3", "goldenrod2", "thistle2", "gray80", "gray80"),   
 labels = korm\_ell\_labels)+  
 xlab("Év") +  
 ylab("Jaccard hasonlóság")+  
 scale\_x\_continuous(breaks=c(unique(tv\_tvjav\_minta$év)), labels=c(unique(tv\_tvjav\_minta$év)))+  
 guides(color=guide\_legend(ncol=3))+  
 theme(axis.text.x = element\_text(angle = 45), legend.position = "bottom")+  
 labs(color = "Beterjesztő")

Mivel a törvényjavaslatok túlnyomó többségét a kabinet tagjai nyújtják be, nem igazán tudunk érdemi következtetéseket levonni arra vonatkozóan, hogy az ellenzéki vagy a kormánypárti képviselők által benyújtott javaslatok módosulnak többet a vita folyamán. Amennyiben ezzel a kérdéssel alaposabban is szeretnénk foglalkozni, érdemes csak azokat a sorokat kiválasztani a hasonlóság-számításhoz, melyben a számunkra releváns megfigyelések szerepelnek. Ha azonban ezt az eljárást választjuk, mindenképpen fontos odafigyelni arra is, hogy az elemzésben használandó megfigyelések kiválogatása nehogy szelektív legyen egy nem megfigyelt változó szempontjából is, ezzel befolyásolva a kutatás eredményeit.

# 11 Természetes-nyelv feldolgozás (NLP) és névelemfelismerés (NER)

A természetes-nyelv feldolgozása (*Natural Language Processing, NLP*) a nyelvészet és a mesterséges intelligencia közös területe, amely a számítógépes módszerek segítségével elemzi az emberek által használt (természetes) nyelveket. Azaz képes feldolgozni különböző szöveges dokumentumok tartalmát, kinyerni a bennük található információkat, kategorizálni és rendszerezni azokat. Angol nyelvű szövegek NLP elemzésére több R csomag is rendelkezésünre áll, ezek közül kettőt mutatunk be röviden. Mivel magyar nyelvű szövegek NLP elemzésre ezek a csomagok jelenleg nem alkalmasak bemutatjuk, hogyan végezhetjük el a magyar nyelvű szövegek mondatra és szavakra bontását, szófaji egyértelműsítését, morfológiai és szintaktikai elemzését az R program használata nélkül és azután a kapott fájlokkal hogyan végezhetünk az R program segítségével további elemzéseket.

A fejezeben részletesen foglalkozunk a névelem-felismeréssel (*Named Entity Recognition, NER*). Névelemnek azokat a tokensorozatokat nevezzük, amelyek valamely entitást egyedi módon jelölnek. A névelem-felismerés az infomációkinyerés részterülete, melynek lényege, hogy automatikusan felismerjük a struktúlálatlan szövegben szereplő tulajdonneveket, majd azokat kigyűjtsük, és típusonként (pl. személynév, földrajzi név, márkanév, stb.) csoportosítsuk. Bár a tulajdonnevek mellett névelemnek tekinthető még például a telefonszámok vagy az email címek is, a névelem-felismerés leginkább mégis a tulajdonnevek felismerésére irányul. A névelem-felismerés a számítógépes nyelvészetben a korai 1990-es évektől kezdve fontos feladatnak és megoldandó problémának számít.A névelem-felismerés többféle módon is megoldható, így péládul felügyelt tanulással, szótár alapú módszerekkel vagy kollokációk elemzésével. A névelem-felismerés körében két alapvető módszer alkalmazására van lehetőség. A szabályalapú módszer alkalmazása során előre megadott adatok alapján kerül kinyerésre az információ (ilyen szabály például a mondatközi nagybetű mint a tulajdonnév kezdete). A másik módszer a statisztikai tanulás, amikor a gép alkot szabályokat a kutató előzetes mintakódolása alapján. A névelemfelismerés során nehézséget okozhat a különböző névelemosztályok közötti gyakori átfedés, így például ha egy adott szó településnév és vezetéknév is lehet. A magyar nyelvű szövegekben a tulajdonnevek automatikus annotációjára jelenleg három módon van lehetőség: tulajdonnév-felismerő algoritmussal, szófaji címke szintjén történő megkülönböztetéssel, valamint szintaktikai szintű címkézéssel. Utóbbi kettőre példa a fejezetben is bemutatásra kerülő magyarlanc elemző, ami szófaji szinten megkülönbözteti a tulajdonneveket, a szintaxis szintjén pedig jelöli a többtagúakat.([Zsibrita, Vincze, and Farkas 2013](#X7820c615428258b8a48ea5bf60d2c7c60b74fda)) A tulajdonnév-felismerő algoritmusok megkeresik az adott szövegben a tulajdonneveket, majd azokat valamilyen kategóriába sorolják, ilyen magyar nyelvű algoritmus a a szeged ner, melynek alkalmazását szintén bemutatjuk.([Szarvas, Farkas, and Kocsor 2006](#ref-szarvasMultilingualNamedEntity2006)) Fontos különbséget tenni a névelem-felismerés és a tulajdonnév-felismerés között. A névelem-felismerésbe beletartozik minden olyan kifejezés amely a világ valamely entitására egyedi módon (unikálisan) referál. Ezzel szemben a tulajdonnév-felismerés, kizárólag a tulajdonnevekre koncentrál.([Üveges 2019](#ref-uvegesNamedEntityRecognition2019); [Vincze 2019](#X80f68af2460a11d06da1d613dd699a9a4494186))

A magyarlancnyelvi előfeldolgozó eszköz a Szegedi Tudományegyetem fejlesztése,([Zsibrita, Vincze, and Farkas 2013](#X7820c615428258b8a48ea5bf60d2c7c60b74fda)) ami magyar nyelvű txt formátumú fájlokat feldolgozva képes egy szöveg mondatokra és szavakra bontására, a szavak morfológiai elemzésére, szófaji egyértelműsítésére, emellett kétféle szintaktikai elemzést is képes hozzárendelni a mondatokhoz. A magyarlanc elérhető a <https://rgai.inf.u-szeged.hu/magyarlanc> oldalon, az innen letölthető jar fájl segítségével a txt formátumú szövegfájlok elemzése parancssorból lehetséges.

A magyarlanchoz hasonlóan az UDPipe nevű elemző szintén képes magyar nyelvű nyers szövegek mondatra és szavakra bontására és szófaji elemzésére, azaz POS-taggelésére (*Part of Speech-tagging*) továbbá a mondatok függőségi elemzésére. Ez az elemző a Universal Dependencies nemzetközileg elismert annotációs sémán alapul.([Straka and Straková 2017](#ref-strakaTokenizingPosTagging2017)) A két nyelvi elemző hasonló funkcionalitásokkal rendelkezik az UDPipe technikailag könnyebben kezelhető, azonban kevésbé pontos elemzési eredményt, mivel jóval kisebb tanitó anyagon lett betanítva mint a magyarlanc.[[42]](#footnote-193)

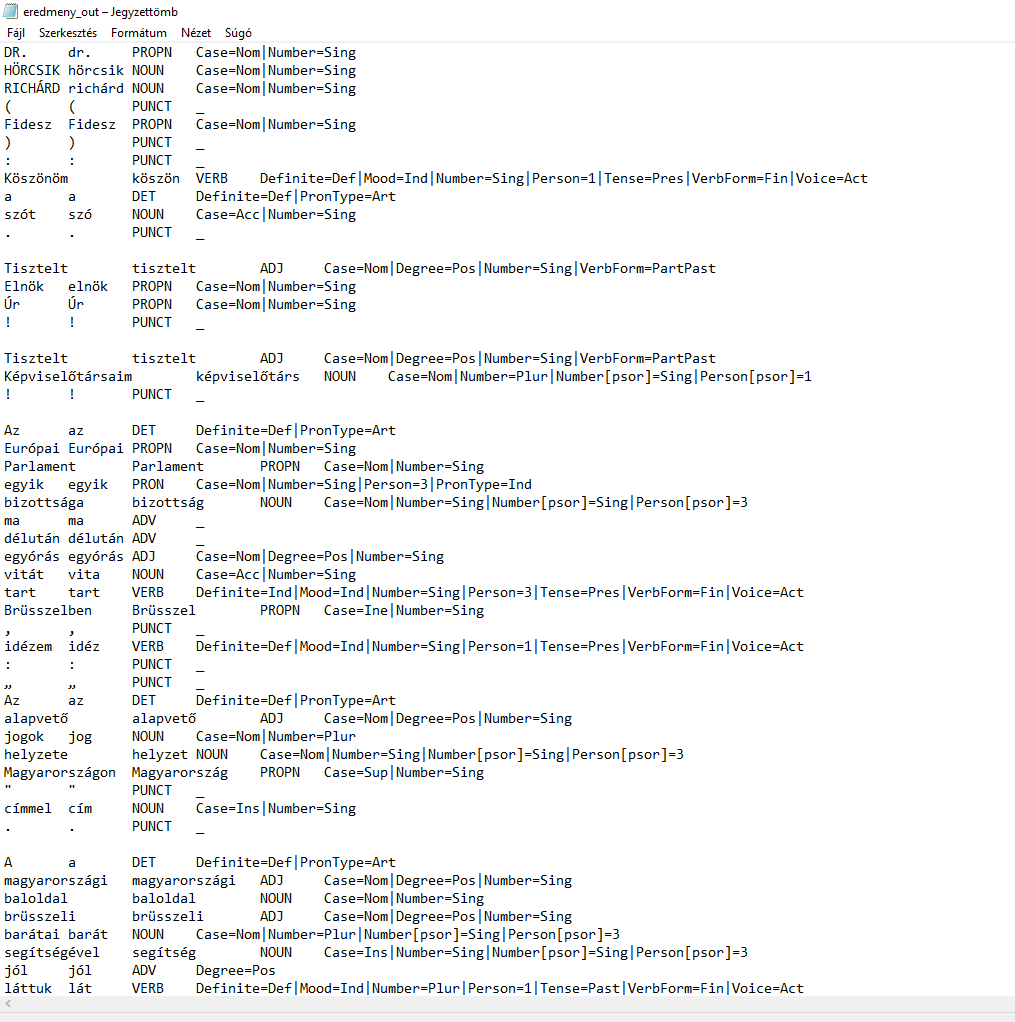
Az alábbiakban a magyarlanc és a a szeged ner működését és az álatuk létrehozott fájlokkal R-ben végezhető elemzésekre mutatunk példákat.

## 11.1 A magyarlanc nyelvi elemző használata

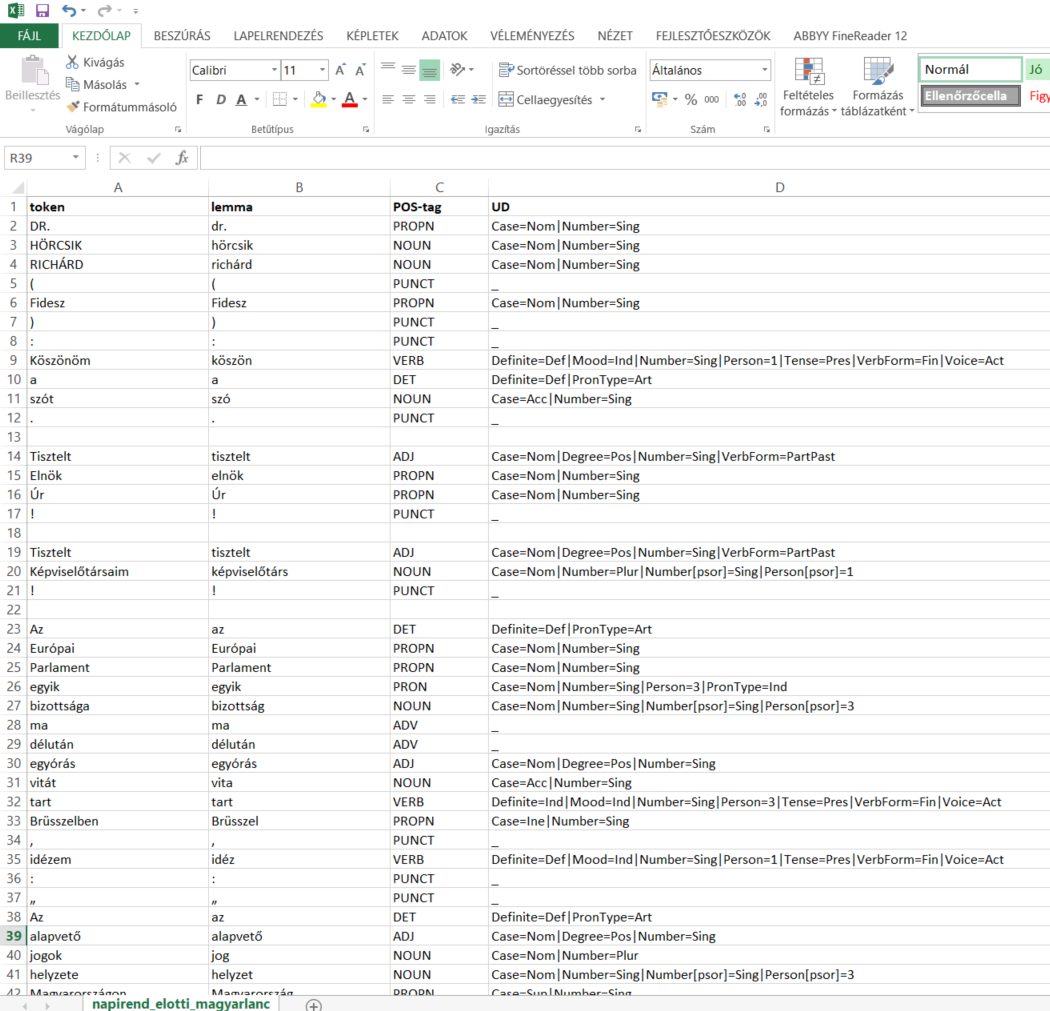
Az elemző használatának részletes leírás megtalálható az [https://rgai.inf.u-szeged.hu/magyarlanc](ttps://rgai.inf.u-szeged.hu/magyarlanc) oldalon, itt most csak vázlatosan ismeretjük. Fontos kiemelni, hogy a magyarlanc JAVA modulokból áll, így használatához szükséges, hogy a számítógépen megfelelő JAVA környezet legyen telepítve. Először fenti oldalról lekell töltenünk a magyarlanc-3.0.jar fájlt, majd bemásolni azt a abba a mappába, ahol az elemezni kívánt txt található. A parancssort Windows operációs rendszer alatt a számítógép kereső mezőjébe a cmd parancs beírásával tudjuk megnyitni. Ezután a parancsorban belépve abba a könyvtárba, ahol az elemezni kíván txt és a magyarlanc-3.0.jar elemző van, az alábbi parancs segítségével végezhetjük el az elemzést: java -Xmx1G -jar magyarlanc-3.0.jar -mode morphparse -input in.txt -output out.txt. Ahol az in.txt helyébe az elemezni kívánt txt nevét, az out.txt helyébe, pedig az elemzés eredményeként létrejövő fájl nevét kell megadni.

Példánkban a Magyar Országgyűlésben 2014 és 2018 között elhangzott véletlenszerűen kiválasztott 25 napirend előtti felszólalás korpuszán szemléltetjük az elemző működését. A 25 fájlt elemezhetjük egyesével, de ha ez a későbbi elemzéshez nem szükséges, a parancsorban a copy \*.txt eredmeny.txt paranccsal egyesíthetjük azokat egy fájlba. Majd ezen az eredmeny.txt-n végezzük el az elemzést az alábbi paranccsal: java -Xmx1G -jar magyarlanc-3.0.jar -mode morphparse -input eredmeny.txt -output eredmeny\_out.txt

Az elemzés eredméynül kapott txt fájlban láthatjuk, hogy az elemző elvégezte a szövegek mondatokrabontását, tokenizálását, lemmatizálását és POS-taggelését, azaz meghatározta a szavak szófaját.



Ezt követően célszerű a txt fájlt excelbe beolvasva oszlopokra tagolni, az oszlopokat fejléccel ellátni, majd csv fájlként elmenteni.



Az így létrehozott csv fájlt ezután a korábban már megismert módon olvashatjuk be a readr csomag segítségével.

library(readr)  
library (dplyr)

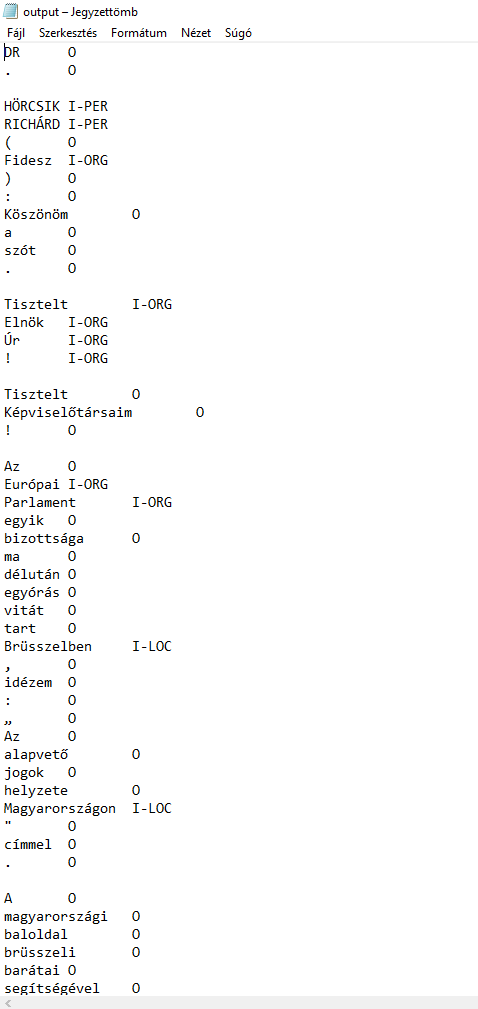
napirend\_elotti <- read\_delim("data/napirend\_elotti\_magyarlanc.csv", delim = ";")

Az így létrehozott data frame objektummal, mely esetünkben 17870 megfigyeést és 4 változót tartalmaz, ezután különböző műveleteket végezhetünk, a korábban már bemutatottak szerint, például dplyr csomag filter függvénye segítségével kiválgathatjuk az igéket, és elmenthetjük azokat egy újabb 1769 megfigyelést és 4 váltiozót tartalmazó objektumba.

verb\_napirend\_elotti <- napirend\_elotti %>%  
 filter(POS\_tag == "VERB")

## 11.2 A szeged ner elemző használata

A magyarlanc nyelvi elemzőhöz hasonlóan használhatjuk a szeged ner elemzőt is, melynek részletes leírása maga a ner.jar elemző is megtalálható az alábbi oldalon <https://rgai.inf.u-szeged.hu/node/109>. Az elemző a fent bemutatott módon szintén parancssorból indítható az alábbi parancs használatával: java -Xmx3G -jar ner.jar -mode predicate -input input.txt -output output.txt Az elemző PER (személynév), LOC (hely(szín)), ORG (szervezet) és MISC (vegyes) címkét ad az egyes névelemeknek.



A fentiekhez hasonlóan ezt a txt-t is átalakíthatjuk táblázattá, majd ezt a csv fájlt beolvashatjuk.

napirend\_elotti\_ner <- read\_delim("data/ner.csv", delim = ";")

Ezután pedig tetszőlegesen kiválogathatjuk például a helyek neveit. A filterezés eredményeként láthatjuk, hogy az elemző korpuszunkban 175 szót azonosított és címkézett fel helynévként.

loc\_napirend\_elotti <- napirend\_elotti\_ner %>%  
 filter(ner == "I-LOC")

De ugyanígy kiváogathatjuk a személyneveket is, azonban itt figyelembe kell vennünk, hogy az elemző külön névelemként jelöli a vezeték és keresztneveket, a további elemzés szükségletei szerint ezeket utólag kell összevonnunk.

pers\_napirend\_elotti <- napirend\_elotti\_ner %>%  
 filter(ner == "I-PER")

Az így kiválogatott különböző névelemekkel azután további elemzéseket végezhetünk.

## 11.3 Angol nyelvű szövegek névelemfelismerése

Amennyiben angol nyelvű korpusszal dolgozunk több lehetőség is a rendelkezésünkre áll a névelemfelsimerés elvégzésére.[[43]](#footnote-202) Amelyek közül most röviden a spacyr használatát mutatjuk be.[[44]](#footnote-204) A spaCy nem egy R, hanem egy Phyton csomag[[45]](#footnote-206), amely azonban az R reticulate csomag segítségével nagyon jól együttműködik a kötetben rendszeresen használt quanteda csomaggal. Használatához a már megszokott módon installálnunk kell a spacyr csomagot, majd beolvasnunk és telepítenünk az angol nyelvi modellt. A Python-ban készült spacy-t a spacyr::spacy\_install() paranccsal kell telepíteni (ezt elég egyszer megtenni, amikor először használjuk a csomagot).

library(spacyr)  
library(quanteda)  
library(ggplot2)  
library(HunMineR)  
  
spacy\_initialize(model = "en\_core\_web\_sm")

A spacy\_parse() függvény segítségével lehetőségünk van a szövekek tokenizálására, lemmatizálsára és POS-taggelésére.

txt <- c(d1 = "spaCy is great at fast natural language processing.",  
 d2 = "Mr. Smith spent two years in North Carolina.")  
  
# process documents and obtain a data.table  
parsedtxt <- spacy\_parse(txt)  
  
parsedtxt  
#> doc\_id sentence\_id token\_id token lemma pos entity  
#> 1 d1 1 1 spaCy spacy NOUN   
#> 2 d1 1 2 is be AUX   
#> 3 d1 1 3 great great ADJ   
#> 4 d1 1 4 at at ADP   
#> 5 d1 1 5 fast fast ADJ   
#> 6 d1 1 6 natural natural ADJ   
#> 7 d1 1 7 language language NOUN   
#> 8 d1 1 8 processing processing NOUN   
#> 9 d1 1 9 . . PUNCT   
#> 10 d2 1 1 Mr. Mr. PROPN   
#> 11 d2 1 2 Smith Smith PROPN PERSON\_B  
#> 12 d2 1 3 spent spend VERB   
#> 13 d2 1 4 two two NUM DATE\_B  
#> 14 d2 1 5 years year NOUN DATE\_I  
#> 15 d2 1 6 in in ADP   
#> 16 d2 1 7 North North PROPN GPE\_B  
#> 17 d2 1 8 Carolina Carolina PROPN GPE\_I  
#> 18 d2 1 9 . . PUNCT

Az elvégzett tokenizálás eredményéből data frame-t készíthetünk.

spacy\_tokenize(txt, remove\_punct = TRUE, output = "data.frame") %>%  
 tail()  
#> doc\_id token  
#> 11 d2 spent  
#> 12 d2 two  
#> 13 d2 years  
#> 14 d2 in  
#> 15 d2 North  
#> 16 d2 Carolina

Ugyancsak lehetőségünk van a különböző entitások, így például a tulajdonnevek kinyerésére

parsedtxt <- spacy\_parse(txt, lemma = FALSE, entity = TRUE, nounphrase = TRUE)  
entity\_extract(parsedtxt)  
#> doc\_id sentence\_id entity entity\_type  
#> 1 d2 1 Smith PERSON  
#> 2 d2 1 North\_Carolina GPE

De a tulajdonneveken túl felcímkézhetjük a dátumokat, eseményeket is.

entity\_extract(parsedtxt, type = "all")  
#> doc\_id sentence\_id entity entity\_type  
#> 1 d2 1 Smith PERSON  
#> 2 d2 1 two\_years DATE  
#> 3 d2 1 North\_Carolina GPE

Az entity\_consolidate() függvény segítségével arra is ehetőségünk van, hogy a több szóból álló entitásokat egy tokenként kezeljük.

entity\_consolidate(parsedtxt) %>%  
 tail()  
#> doc\_id sentence\_id token\_id token pos entity\_type  
#> 11 d2 1 2 Smith ENTITY PERSON  
#> 12 d2 1 3 spent VERB   
#> 13 d2 1 4 two\_years ENTITY DATE  
#> 14 d2 1 5 in ADP   
#> 15 d2 1 6 North\_Carolina ENTITY GPE  
#> 16 d2 1 7 . PUNCT

A nounphrase\_extract() függvény lehetőséget ad az összetartozó kifejezések összefűzésére.

nounphrase\_extract(parsedtxt)  
#> doc\_id sentence\_id nounphrase  
#> 1 d1 1 spaCy  
#> 2 d1 1 fast\_natural\_language\_processing  
#> 3 d2 1 Mr.\_Smith  
#> 4 d2 1 two\_years  
#> 5 d2 1 North\_Carolina

Majd képes arra, hogy ezeket az összetartozó kifejezéseket egyben kezelje.

nounphrase\_consolidate(parsedtxt)  
#> doc\_id sentence\_id token\_id token pos  
#> 1 d1 1 1 spaCy nounphrase  
#> 2 d1 1 2 is AUX  
#> 3 d1 1 3 great ADJ  
#> 4 d1 1 4 at ADP  
#> 5 d1 1 5 fast\_natural\_language\_processing nounphrase  
#> 6 d1 1 6 . PUNCT  
#> 7 d2 1 1 Mr.\_Smith nounphrase  
#> 8 d2 1 2 spent VERB  
#> 9 d2 1 3 two\_years nounphrase  
#> 10 d2 1 4 in ADP  
#> 11 d2 1 5 North\_Carolina nounphrase  
#> 12 d2 1 6 . PUNCT

Arra is lehetőség van, hogy az egyes kifejezések közötti függőségeket vizsgáljuk.

spacy\_parse(txt, dependency = TRUE, lemma = FALSE, pos = FALSE)  
#> doc\_id sentence\_id token\_id token head\_token\_id dep\_rel entity  
#> 1 d1 1 1 spaCy 2 nsubj   
#> 2 d1 1 2 is 2 ROOT   
#> 3 d1 1 3 great 2 acomp   
#> 4 d1 1 4 at 2 prep   
#> 5 d1 1 5 fast 8 amod   
#> 6 d1 1 6 natural 7 amod   
#> 7 d1 1 7 language 8 compound   
#> 8 d1 1 8 processing 4 pobj   
#> 9 d1 1 9 . 2 punct   
#> 10 d2 1 1 Mr. 2 compound   
#> 11 d2 1 2 Smith 3 nsubj PERSON\_B  
#> 12 d2 1 3 spent 3 ROOT   
#> 13 d2 1 4 two 5 nummod DATE\_B  
#> 14 d2 1 5 years 3 dobj DATE\_I  
#> 15 d2 1 6 in 3 prep   
#> 16 d2 1 7 North 8 compound GPE\_B  
#> 17 d2 1 8 Carolina 6 pobj GPE\_I  
#> 18 d2 1 9 . 3 punct

A következőkben a “Szótáralapú elemzések, érzelem-elemzés” fejezetben is használt Magyar Nemzeti Bank kamatdöntéseit kísérő nemzetközi sajtóközleményei korpusznán mutatunk be egy példát a névelemfelismerésre és az eredmények vizualizálására.

Első lépésként beolvassuk a szövegeket, majd a már megismert quanteda csomag segítségévek korpuszt készítünk belőlük.

mnb\_df <- data\_mnb\_pr  
  
corpus\_mnb <-corpus(mnb\_df)

Ezután a spacy\_extract\_entity() függvénye segítségévek elvégezzük a névelemfelismerést, a függvény argumnetumában megadva, hogy milyen tipusú névelemekt szeretnénk kigyűjteni a korpuszból. A lehetséges típusok a **named**, **extended**, vagy **all**.[[46]](#footnote-208) Az elemzés eredménye pedig készülhet listában, vagy készülhet belőle data frame. Példánkban mi kimenetként listát állítottunk be.

A névelemek tokenjeit ezután megritkítottuk, és csak azokat hagytuk meg, amelyek legalább nyolc alkalommal szerepltek a korpuszban.

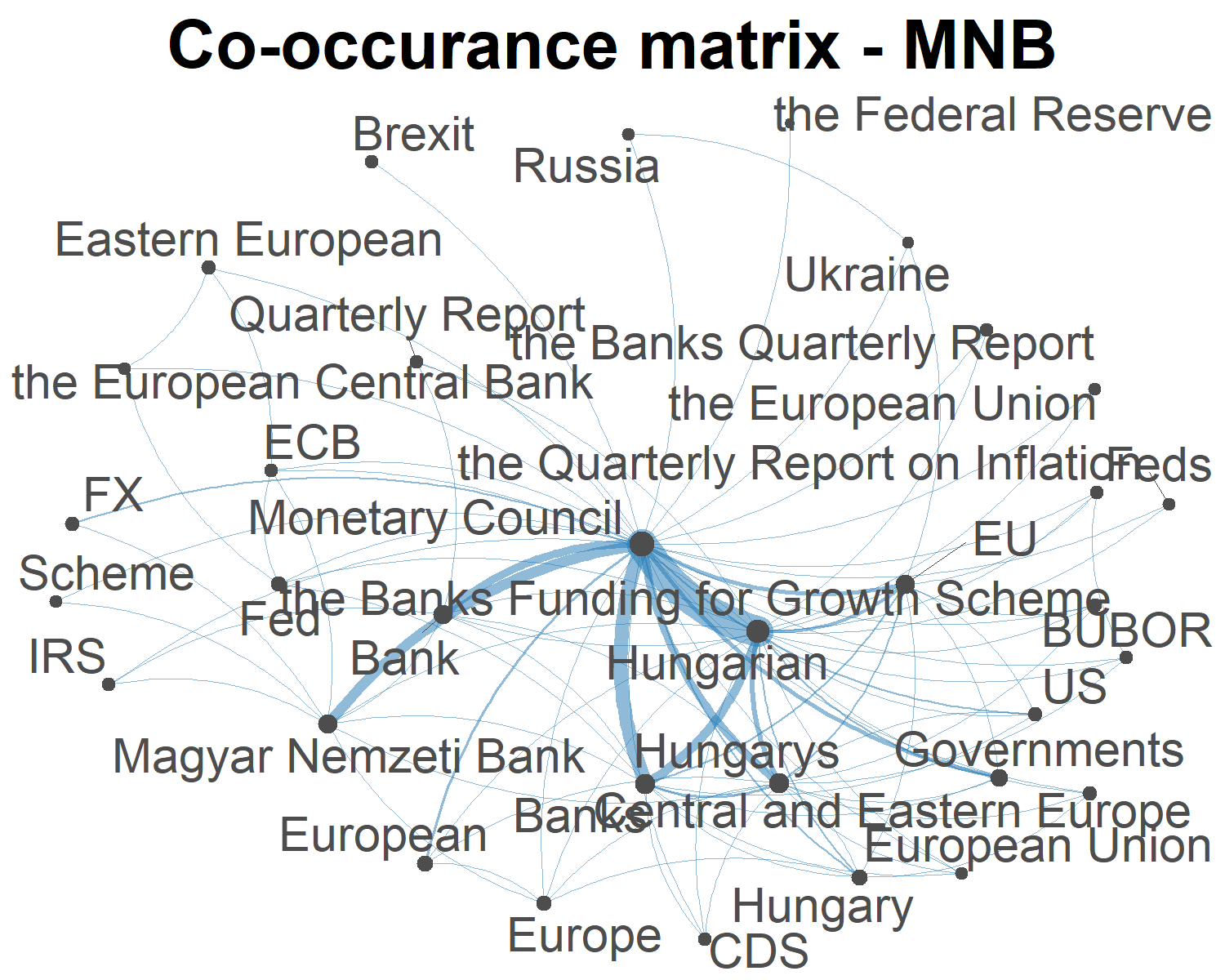
mnb\_ner <- spacy\_extract\_entity(  
 corpus\_mnb,  
 output = c("list"),  
 type = ("named"),  
 multithread = TRUE)  
  
mnb\_tokens <- tokens(mnb\_ner)  
  
features <- dfm(mnb\_tokens) %>%  
 dfm\_trim(min\_termfreq = 8) %>%  
 featnames()  
  
mnb\_tokens <- tokens\_select(mnb\_tokens, features, padding = TRUE)

Ezután a különböző alakban előforduló, de ugyanarra az entitásra vonatkozó névelemeket összevontuk.

mnb <- c("Magyar Nemzeti Bank", "MAGYAR NEMZETI BANK", "The Magyar Nemzeti Bank ", "the Magyar Nemzeti Bank", "MNB", "the Magyar Nemzeti Banks", "Nemzeti Bank", "The Magyar Nemzeti Banks MNB" )  
lemma <- rep("Magyar Nemzeti Bank", length(mnb))  
mnb\_tokens <- tokens\_replace(mnb\_tokens, mnb, lemma, valuetype = "fixed")  
  
mc <- c("Monetary Council", "MONETARY COUNCIL", "Magyar Nemzeti Bank Monetary Council", "MAGYAR NEMZETI BANK Monetary Council", "NEMZETI BANK Monetary Council" ,"The Monetary Council", "Council", "The Council", "Councils", "the Monetary Council", "Monetary Councils", "the Monetary Councils", "The Monetary Councils", "Monetray Council", "May the Monetary Council")  
lemma <- rep("Monetary Council", length(mc))  
mnb\_tokens <- tokens\_replace(mnb\_tokens, mc, lemma, valuetype = "fixed")

Majd elkészítettük a szóbeágyazás fejezetben már megismert fcm-et, végezetül pedig egy együttes előforulási mátrixot készítettünk a kinyert entitásokból és a ggplot segítségével ábrázoltuk.[[47]](#footnote-210)

mnb\_fcm <- fcm(mnb\_tokens, context = "window", count = "weighted", weights = 1 / (1:5), tri = TRUE)  
  
feat <- names(topfeatures(mnb\_fcm, 80))  
mnb\_fcm\_select <- fcm\_select(mnb\_fcm, pattern = feat, selection = "keep")  
dim(mnb\_fcm\_select)  
#> [1] 35 35  
  
size <- log(colSums(dfm\_select(mnb\_fcm, feat, selection = "keep")))  
  
set.seed(144)  
  
textplot\_network(mnb\_fcm\_select, min\_freq = 0.7, vertex\_size = size / max(size) \* 3)+  
 ggtitle("Co-occurance matrix - MNB")+  
 theme(plot.title = element\_text(hjust = 0.5))+  
 theme(plot.title = element\_text(size = 20, face = "bold"))



A spacyr alapvetően az angol nyelvi modellel működik, de arra is van lehetőség, hogy a spaCy egyéb beépített nyelvi modelljeit (német, spanyol, portugál) használjuk. Létezik magyar nyelvi modell is, ez azonban jelenleg még nincs integrálva a spaCy-be, hanem egy GitHub repozitoriból tölthető le. Ennek R-ben történő megvalósításához azonban haladó R ismeretek szükségesek, azért ennek leírásától jelen kötetben eltekintünk. A magyar nyelvi modell és leírása elérhető az alábbi linken: <https://github.com/oroszgy/spacy-hungarian-models>

# 12 Osztályozás és felügyelt tanulás

tizenkeddik fejezet

# 13 Függelék

## 13.1 Az R és az RStudio használata

Az R egy programozási nyelv, amely alkalmas statisztikai számítások elvégzésére és ezek eredményeinek grafikus megjelenítésére. Az R ingyenes, nyílt forráskódú szoftver, mely telepíthető mind Windows, mind Linux, mind MacOS operációs rendszerek alatt, az alábbi oldalról: <https://cran.r-project.org/> Az RStudio az R integrált fejlesztői környezete (*integrated development environment, IDE*), mely egy olyan felhasználóbarát felületet biztosít, ami egyszerűbb és átláthatóbb munkát tesz lehetővé. Az RStudio az alábbi oldalról tölthető le: <https://rstudio.com/products/rstudio/download/>

A *„point and click"* szoftverekkel szemben az R használata során scripteket kell írni, ami bizonyos programozási jártasságot feltételez, de a későbbiekben lehetővé teszi azt adott kutatási kérdéshez maximálisan illeszkedő kódok összeállítását, melyek segítségével az elemzések mások számára is megbízhatóan reprodukálhatók lesznek. Ugyancsak az R használata mellett szól, hogy komoly fejlesztői és felhasználói közösséggel rendelkezik, így a használat során felmerülő problémákra általában gyorsan megoldást találhatunk.

### 13.1.1 Az RStudio kezdőfelülete

Az RStudio kezdőfelülete négy panelből, eszközsorból és menüsorból áll:

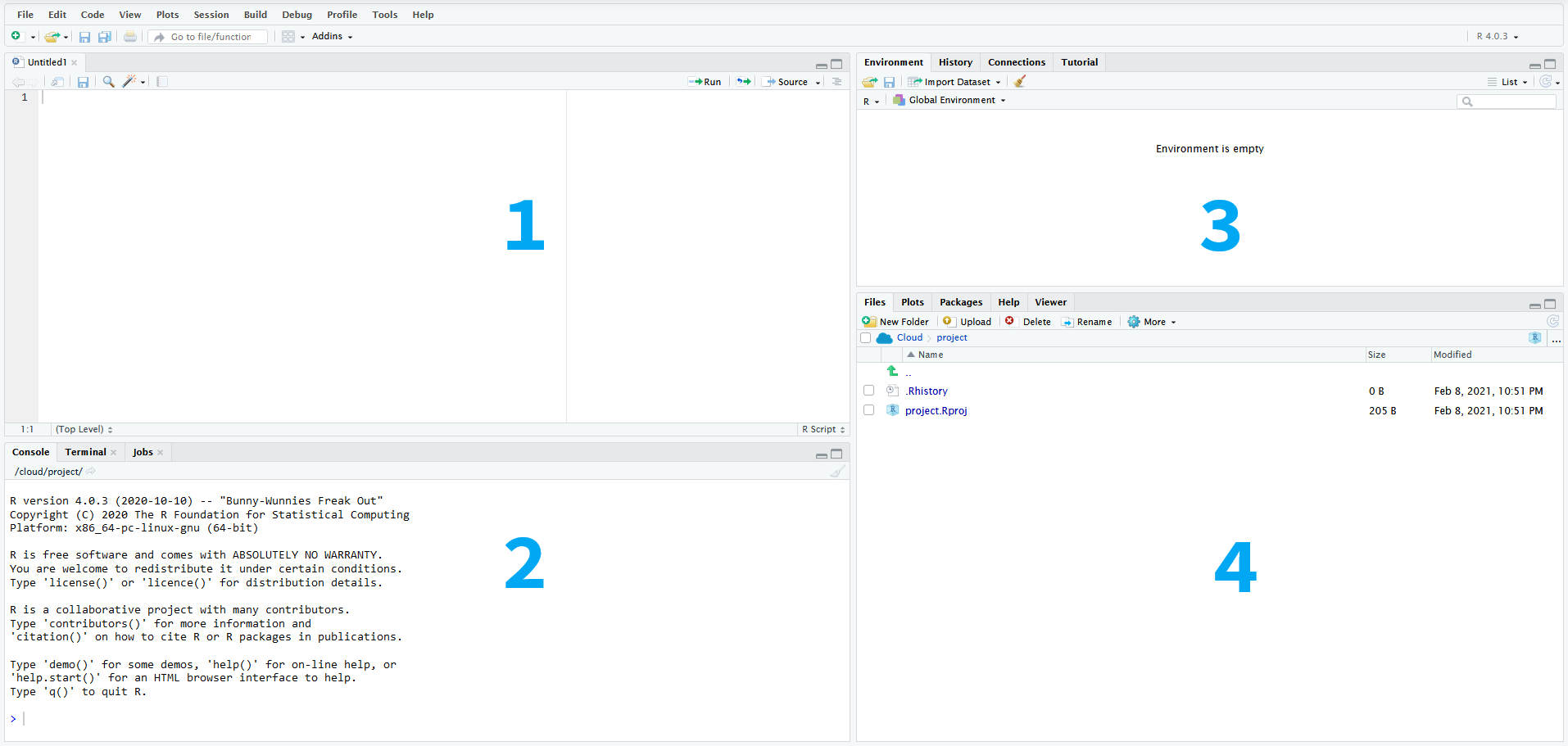


Figure 13.1: RStudio felhasználói felület

Az **(1) editor** ablak szolgál a kód beírására, futtatására és mentésére. A **(2) console** ablakban jelenik meg a lefuttatott kód és az eredmények. A jobb felső ablak **(3) environment** fülén láthatóak a memóriában tárolt adatállományok, változók és felhasználói függvények. A **history** fül mutatja a korábban lefuttatott utasításokat. A jobb alsó ablak **(4) files** fülén az aktuális munkakönyvtárban tárolt mappákat és fájlok találjuk, míg a **plot** fülön az elemzéseink során elkészített ábrák jelennek meg. A **packages** fülön frissíthetjük a meglévő r csomagokat és telepíthetünk újakat. A **help** fülön a különböző függvények, parancsok leírását, és használatát találjuk meg. A Tools -> Global Options menüpont végezhetjük el az RStudio testreszabását. Így például beállíthatjuk az ablaktér elrendezését (*Pane layout*), vagy a színvilágot (*Appearance*), illetve azt hogy a kódok ne fussanak ki az ablakból (Code -> Editing -> Soft wrap R source files)

### 13.1.2 Projekt alapú munka

Bár nem kötelező, de javasolt, hogy az RStudio-ban projekt alapon dolgozzunk, mivel így az összes – az adott projekttel kapcsolatos fájlt – egy mappában tárolhatjuk. Új projekt beállítását a File->New Project menüben tehetjük meg, ahol a saját gépünk egy könyvtárát kell kiválasztani, ahová az R a scripteket, az adat- és előzményfájlokat menti. Ezenkívül a Tools->Global Options->General menüpont alatt le kell tiltani a *„Restore most recently opened project at startup”* és a *„Restore .RData ino workspace at startup”* beállítást, valamint *„Save workspace to .RData on exit”* legördülő menüjében be kell állítani a *„Never”* értéket.

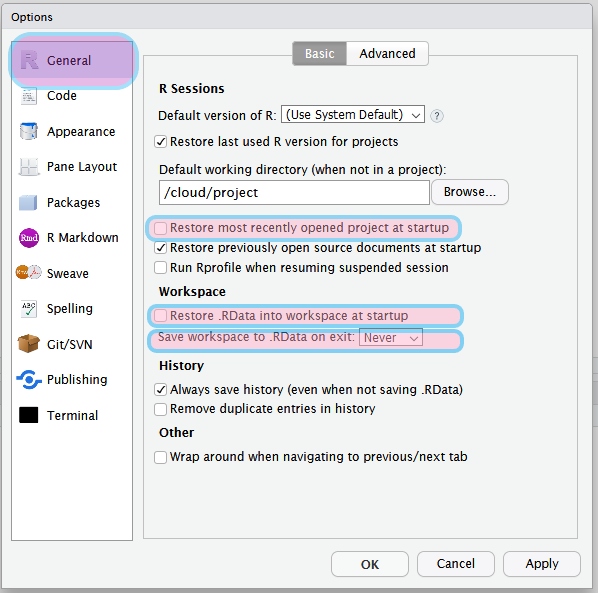


Figure 13.2: RStudio projekt beállítások

A szükséges beállítások után a File -> New Project menüben hozhatjuk létre a projektet. Itt arra is lehetőségünk van, hogy kiválasszuk, hogy a projektünket egy teljesen új könyvtárba, vagy egy meglévőbe kívánjuk menteni, esetleg egy meglévő projekt új verzióját szeretnénk létrehozni. Ha sikeresen létrehoztuk a projektet, az RStudio jobb felső sarkában látnunk kell annak nevét.

### 13.1.3 Scriptek szerkesztése, függvények használata

Új script a File -> New -> File -> R Script menüpontban hozható létre, mentésére a File->Save menüpontban egy korábbi script megnyitására File -> Open menüpontban van lehetőségünk. Script bármilyen szövegszerkesztővel írható, majd beilleszthető az **editor** ablakba. A scripteket érdemes magyarázatokkal (kommentekkel) ellátni, hogy a későbbiekben pontosan követhető legyen, hogy melyik parancs segítségével pontosan milyen lépéseket hajtottunk végre. A magyarázatokat vagy más néven kommenteket kettőskereszt (#) karakterrel vezetjük be. A scriptbeli utasítások az azokat tartalmazó sorokra állva vagy több sort kijelölve a Run feliratra kattintva vagy a Ctrl+Enter billentyűparanccsal futtathatók le. A lefuttatott parancsok és azok eredményei ezután a bal alsó sarokban lévő **console** ablakban jelennek meg és ugyanitt kapunk hibaüzenetet is, ha valamilyen hibát vétettünk a script írása közben.

A munkafolyamat során létrehozott állományok (ábrák, fájlok) az ún. munkakönyvtárba (*working directory*) mentődnek. Az aktuális munkakönyvtár neve, elérési útja a getwd() utasítással jeleníthető meg. A könyvtárban található állományok listázására a list.files() utasítással van lehetőségünk. Ha a korábbiaktól eltérő munkakönyvtárat akarunk megadni, azt a setwd() függvénnyel tehetjük meg, ahol a ()-ben az adott mappa elérési útját kell megadnunk. Az elérési útban a meghajtó azonosítóját, majd a mappák, almappák nevét vagy egy normál irányú perjel (/), vagy két fordított perjel (\\) választja el, mivel az elérési út karakterlánc, ezért azt idézőjelek vagy aposztrófok közé kell tennünk. Az aktuális munkakönyvtárba beléphetünk a jobb alsó ablak file lapján a „More -> Go To Working Directory” segítségével. Ugyanitt a „Set Working Directory”-val munkakönyvtárnak állíthatjuk be az a mappát, amelyben épp benne vagyunk.

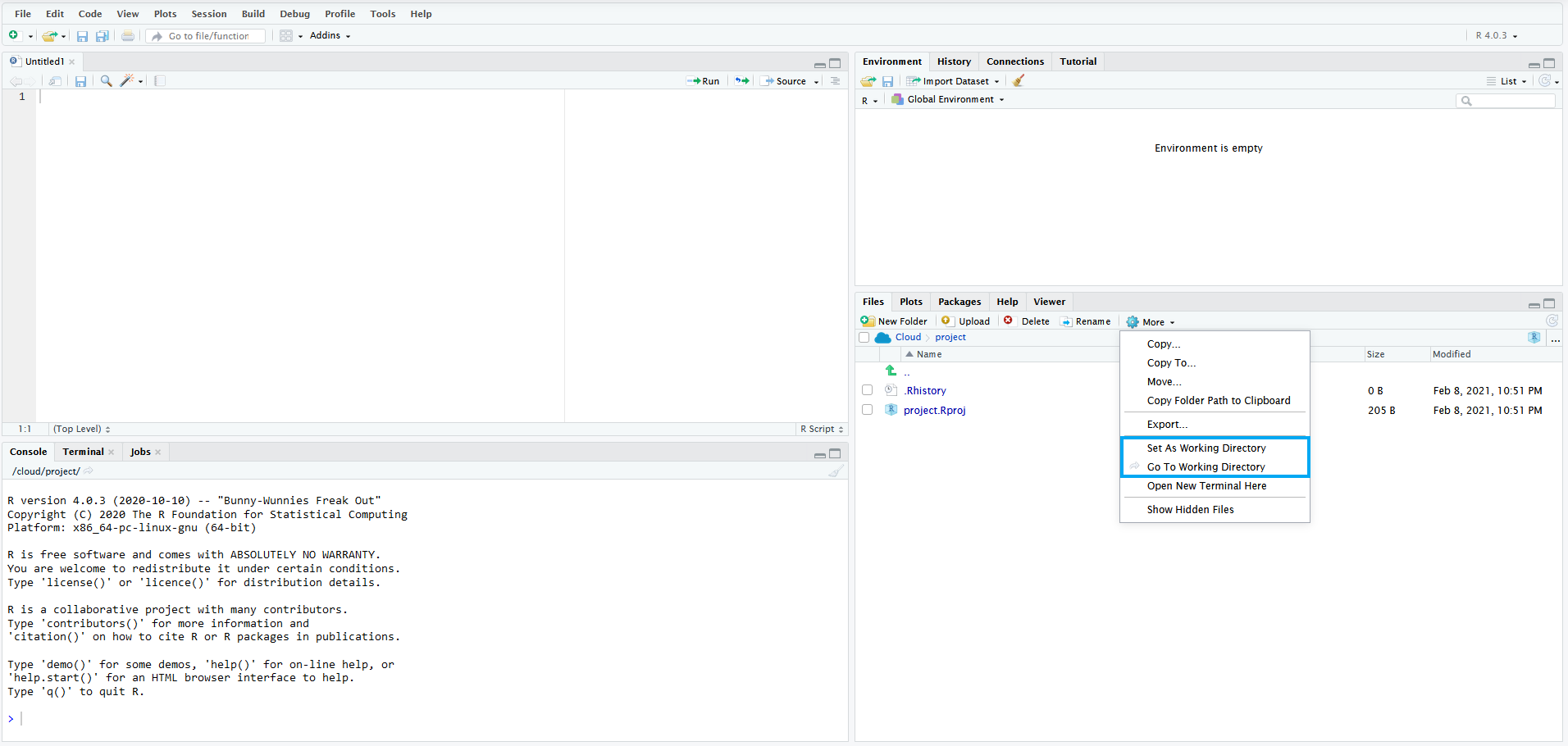


Figure 13.3: Working directory beállítások

A munkafolyamat befejezésére a q() vagy quit() függvénnyel van lehetőségünk. Az R-ben **objektumokkal** dolgozunk, amik a teljesség igénye nélkül lehetnek például egyszerű szám vektortok, vagy akár komplex listák, illetve függvények, ábrák. A munkafolyamat során létrehozott **objektumokat** az RStudio jobb felső ablakának **environment** fülén jelennek meg. A mentett objektumokat a fent látható seprű ikonra kattintva törölhetjük a memóriából. Az **environment** ablakra érdemes úgy gondolni hogy ott jelennek meg a memóriában tárolt értékek.

Az RStudio jobb alsó ablakának **plots** fülén láthatjuk azon parancsok eredményét, melyek kimenete valamilyen ábra. A **packages** fülnél a már telepített és a letölthető kiegészítő csomagokat jeleníthetjük meg. A **help** fülön a korábban említettek szerint a súgó érhető el. Az RStudio-ban használható billentyűparancsok teljes listáját Alt+Shift+K billentyűkombinációval tekinthetjük meg. Néhány gyakrabban használt, hasznos billentyűparancs:

* Ctrl+Enter: futtassa a kódot az aktuális sorban
* Ctrl+Alt+B: futtassa a kódot az elejétől az aktuális sorig
* Ctrl+Alt+E: futtassa a kódot az aktuális sortól a forrásfájl végéig
* Ctrl+D: törölje az aktuális sort

Az R-ben beépített **függvények** (*function*) állnak rendelkezésünkre a számítások végrehajtására, emellett több **csomag** (*package*) is letölthető, amelyek különböző függvényeket tartalmaznak. A függvények a következőképpen épülnek fel: függvénynév(paraméter). Például tartalom képernyőre való kiíratását a print() függvénnyel tehetjük, amelynek gömbölyű zárójelekkel határolt részébe írhatjuk a megjelenítendő szöveget. A citation() függvénnyel lekérdezhetjük az egyes beépített csomagokra való hivatkozást is: a citation(quanteda) függvény a quanteda csomag hivatkozását adja meg. Az R súgórendszere a help.start() utasítással indítható el. Egy adott függvényre vonatkozó súgórészlet a függvények neve elé kérdőjel írásával, vagy a help() argumentumába a kérdéses függvény nevének beírásával jeleníthető meg (pl.: help(sum)).

### 13.1.4 R csomagok

Az R-ben telepíthetők kiegészítő csomagok (packages), amelyek alapértelmezetten el nem érhető algoritmusokat, függvényeket tartalmaznak. A csomagok saját dokumentációval rendelkeznek, amelyeket fel kell tüntetni a használatukkal készült publikációink hivatkozáslistájában. A csomagok telepítésre több lehetőségünk is van: használhatjuk a menüsor Tools -> Install Packages menüpontját, vagy a jobb alsó ablak **packages** fül Install menüpontját, illetve az **editor** ablakban az install.packages() parancsot futtatva, ahol a ()-be a telepíteni kívánt csomag nevét kell beírnunk (pl.: install.packages(dplyr)).

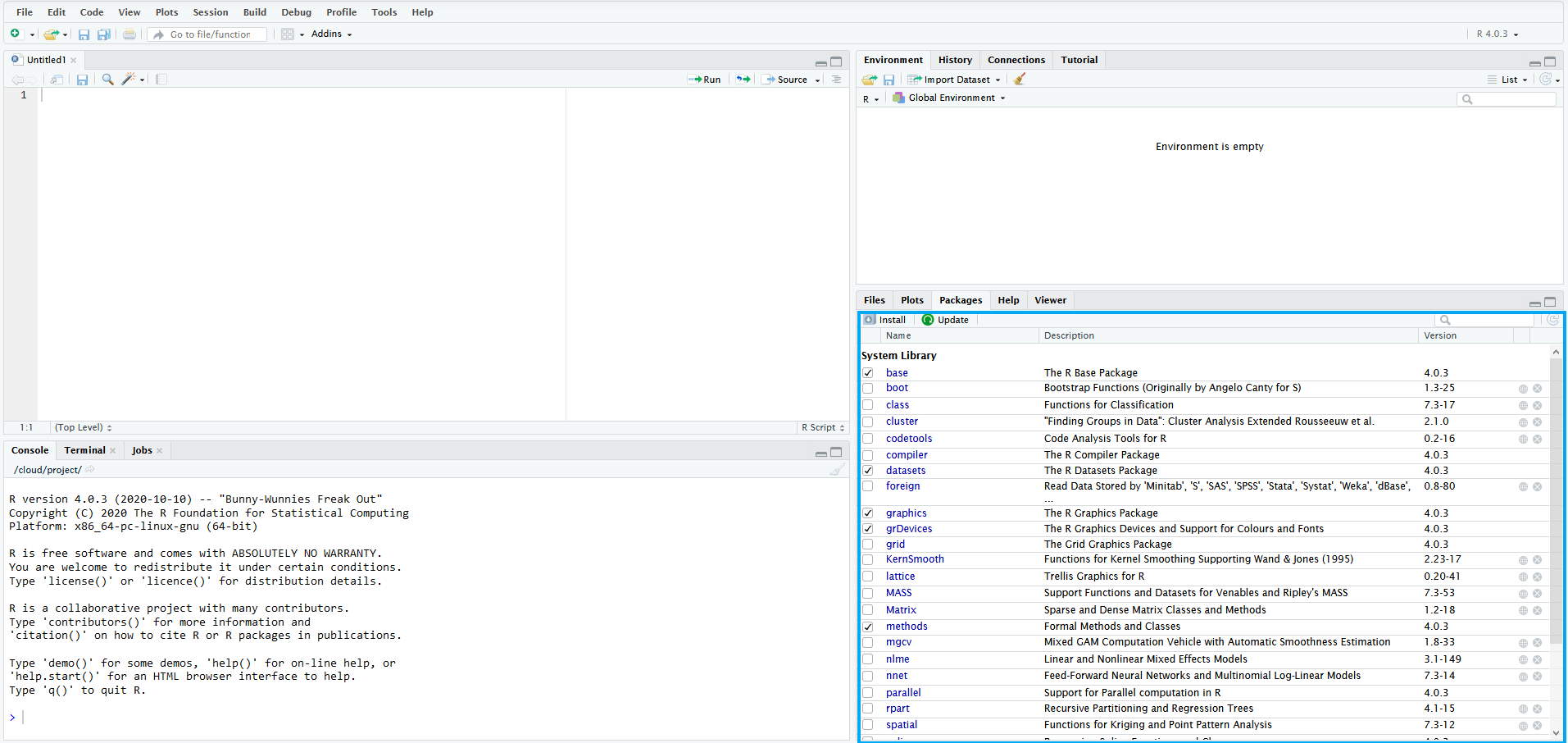


Figure 13.4: Packages fül

### 13.1.5 Objektumok tárolása, értékadás

Az objektumok lehetnek például **vektorok**, **mátrixok**, **tömbök** (*array*), **adat táblák** (*data frame*). Értékadás nélkül az R csak megjeleníti a műveletek eredményét, de nem tárolja el azokat. Az eredmények eltárolásához azokat egy objektumba kell elmentenünk. Ehhez meg kell adnunk az objektum nevét majd az <- után adjuk meg annak értékét: a <- 12 + 3.Futtatás után az environments fülön megjelenik az a objektum, melynek értéke 15. Az objektumok elnevezésénél figyelnünk kell arra, hogy az R különbséget tesz a kis és nagybetűk között, valamint, hogy az ugyanolyan nevű objektumokat kérdés nélkül felülírja és ezt a felülírást nem lehet visszavonni.

### 13.1.6 Vektorok

Az R-ben kétféle típusú vektort különböztetünk meg:

* **egyedüli vektor** (*atomic vector*)
* **lista** (*list*)

Az egyedüli vektornak hat típusa van, **logikai** (*logical*), **egész szám** (*integer*), **természetes szám** (*double*), **karakter** (*character*), **komplex szám** (*complex*) és **nyers adat** (*raw*). A leggyakrabban valamilyen numerikus, logikai vagy karakter vektorral használjuk. Az egyedüli vektorok onnan kapták a nevüket hogy csak egy féle adattípust tudnak tárolni. A listák ezzel szemben gyakorlatilag bármit tudnak tárolni, akár több listát is egybeágyazhatunk.

A vektorok és listák azok az építőelemek amikből felépülnek az R objektumaink. Több érték vagy azonos típusú objektum összefűzését a c() függvénnyel végezhetjük el. A lenti példában három különböző objektumot kreálunk, egy numerikusat, egy karaktert és egy logikait. A karakter vektorban az elemeket időzőjellel és vesszővel szeparáljuk. A logikai vektor csak TRUE, illetve FALSE értékeket tartalmazhat.

numerikus <- c(1,2,3,4,5)  
  
karakter <- c("kutya","macska","ló")  
  
logikai <- c(TRUE, TRUE, FALSE)

A létrehozott vektorokkal különböző műveleteket végezhetünk el, például összeadhatjuk numerikus vektorainkat. Ebben az esetben az első vektor első eleme a második vektor első eleméhez adódik.

c(1:4) + c(10,20,30,40)  
#> [1] 11 22 33 44

A karaktervektorokat össze is fűzhetjük egymással. Példánkban egy új objektumot is létrehoztunk, ezért a jobb felső ablakban, az **environment** fülön láthatjuk, hogy a létrejött karakter\_kombinalt objektum egy négy elemű (hosszúságú) karaktervektor (chr [1:4]), melynek elemei a "kutya","macska","ló","nyúl". Az objektumként tárolt vektorok tartalmát az adott sort lefuttatva írathatjuk ki a **console** ablakba. Ugyanezt megtehetjük print() függvény segítségével is, ahol a függvény arrgumentumában () az adott objektum nevét kell szerepeltetnünk.

karakter1 <- c("kutya","macska","ló")  
karakter2 <-c("nyúl")  
  
karakter\_kombinalt <-c(karakter1, karakter2)  
  
karakter\_kombinalt  
#> [1] "kutya" "macska" "ló" "nyúl"

Ha egy vektorról szeretnénk megtudni, hogy milyen típusú azt a typeof() vagy a class() paranccsal tehetjük meg, ahol ()-ben az adott objektumként tárolt vektor nevét kell megadnunk: typeof(karakter1). A vektor hosszúságát (benne tárolt elemek száma vektorok esetén) a lenght() függvénnyel tudhatjuk meg.

typeof(karakter1)  
#> [1] "character"  
  
length(karakter1)  
#> [1] 3

### 13.1.7 Faktorok

A faktorok a kategórikus adatok tárolására szolgálnak. Faktor típusú változó a factor() függvénnyel hozható létre. A faktor szintjeit (igen, semleges, nem), a levels() függvénnyel kaphatjuk meg míg az adatok címkéit (tehát a kapott válaszok számát), a labels() paranccsal érhetjük el.

survey\_response <- factor(c("igen", "semleges", "nem", "semleges", "nem", "nem", "igen"), ordered = TRUE)  
  
levels(survey\_response)  
#> [1] "igen" "nem" "semleges"  
  
labels(survey\_response)  
#> [1] "1" "2" "3" "4" "5" "6" "7"

### 13.1.8 Data frame

Az adattábla (*data frame*) a statisztikai és adatelemzési folyamatok egyik leggyakrabban használt adattárolási formája. Egy data frame többféle típusú adatot tartalmazhat. A data frame-k különféle oszlopokból állhatnak, amelyek különféle típusú adatokat tartalmazhatnak, de egy oszlop csak egy típusú adatból állhat. Az itt bemutatott *data frame* 7 megfigyelést és 4 féle változót tartalmaz (id, country, pop, continent).

#> id orszag nepesseg kontinens  
#> 1 1 Thailand 68.7 Asia  
#> 2 2 Norway 5.2 Europe  
#> 3 3 North Korea 24.0 Asia  
#> 4 4 Canada 47.8 North America  
#> 5 5 Slovenia 2.0 Europe  
#> 6 6 France 63.6 Europe  
#> 7 7 Venezuela 31.6 South America

A *data frame*-be rendezett adatokhoz különböző módon férhetünk hozzá, például a *data frame* nevének majd []-ben a kívánt sor megadásával, kiírathatjuk a **console** ablakba annak tetszőleges sorát ás oszlopát: orszag\_adatok[1, 1]. Az R több különböző módot kínál a *data frame* sorainak és oszlopainak eléréséhez. A [ általános használata: data\_frame[sor, oszlop]. Egy másik megoldás a $ haszálata: data\_frame$oszlop.

orszag\_adatok[1, 4]  
#> [1] Asia  
#> Levels: Asia Europe North America South America  
  
orszag\_adatok$orszag  
#> [1] "Thailand" "Norway" "North Korea" "Canada" "Slovenia"   
#> [6] "France" "Venezuela"

## 13.2 Vizualizáció

library(ggplot2)  
library(gapminder)

Az elemzéseinkhez használt data frame adatainak alapján a ggplot2 csomag segítségével lehetőségünk van különböző vizualizációk készítésére is.

A ggplot2 használata során különböző témákat alkalmazhatunk, melyek részletes leírása megtalálható: <https://ggplot2.tidyverse.org/reference/ggtheme.html>

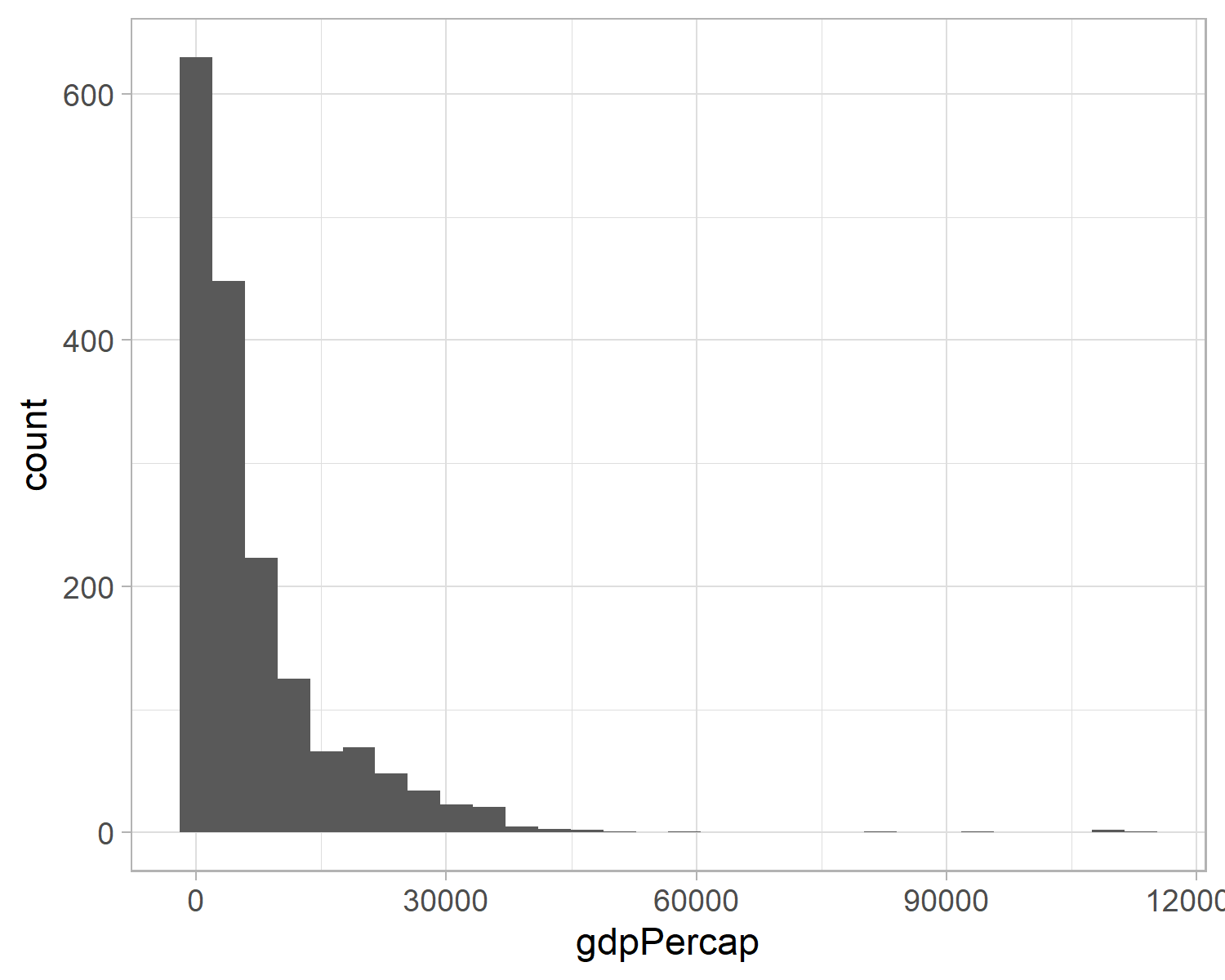
Abban az esetben, ha nem választunk témát, a ggplot2 a következő ábrán is látható alaptémát használja. Ha például a szürke helyett fehér hátteret szeretnénk, alkalmazhatjuk a theme\_minmal()parancsot. Szintén gyakran alkalmazott ábra alap a thema\_bw(), ami az előzőtől az ábra keretezésében különbözik. Ha fehér alapon, de a beosztások vonalait feketén szeretnénk megjeleníteni, alkalmazhatjuk a theme\_linedraw() függvényt, a theme\_void() segítségével pedig egy fehér alapon, beosztásoktól mentes alapot kapunk, a theme\_dark() pedig sötét hátteret eredményez. A theme\_classic() segítségével az x és y tengelyt jeleníthetjük meg fehér alapon.

Egy ábra készítésének alapja mindig a használni kívánt adatkészlet beolvasása, illetve az ábrázolni kívánt változót vagy változók megadása.

Ezt követi a megfelelő alakzat kiválasztása, attól függően például, hogy eloszlást, változást, adatok közötti kapcsolatot, vagy eltéréseket akarunk ábrázolni. A geom az a geometriai objektum, a mit a diagram az adatok megjelenítésére használ. Agglpot2 több mint 40 féle alakzat alkalmazására ad lehetőséget, ezek közül néhány gyakoribbat mutatunk be az alábbiakban. Az alakzatokról részletes leírása található például az alábbi linken: <https://r4ds.had.co.nz/data-visualisation.html>

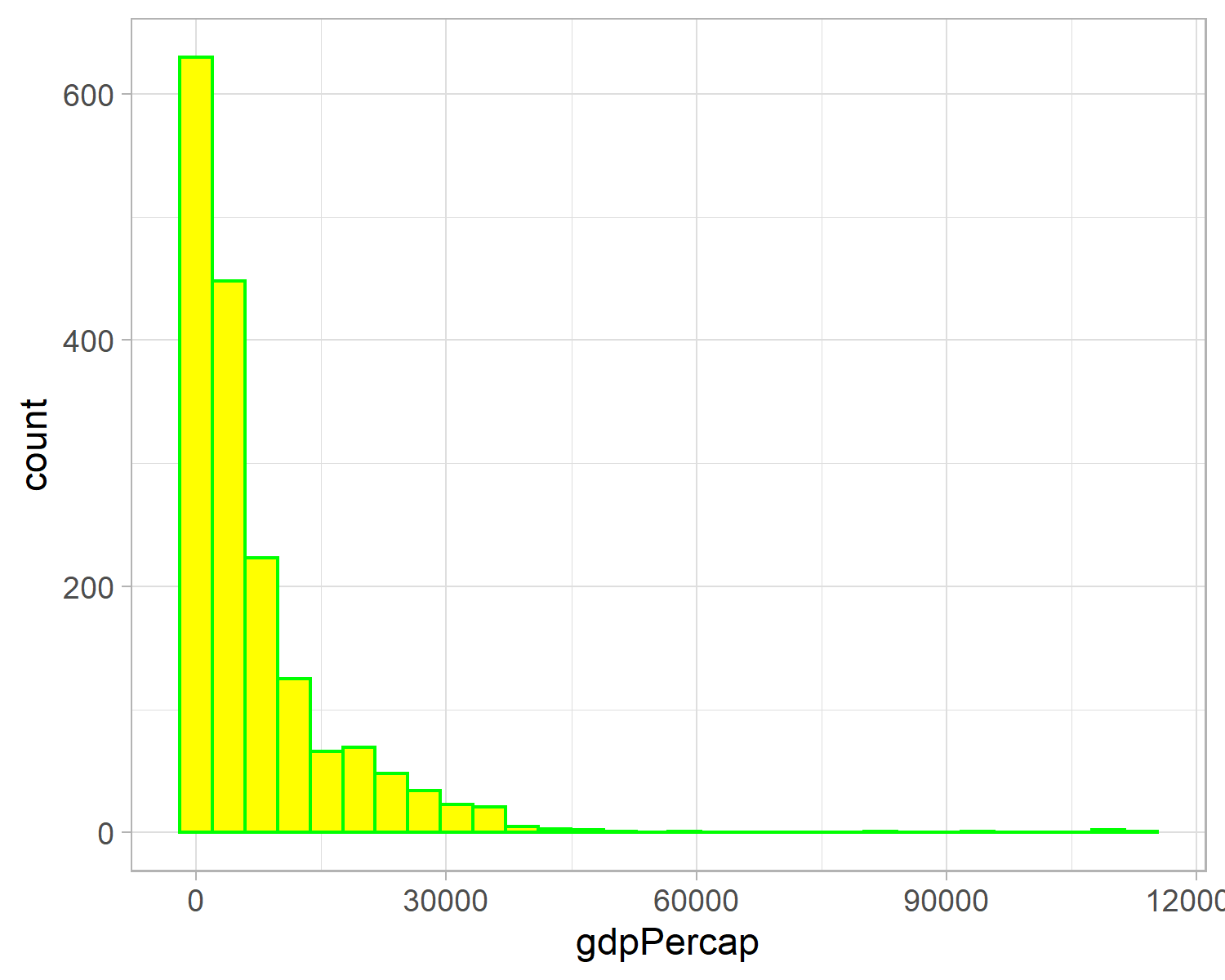
A következőkben a gapminder csomagban található adatok segítségével szemléltetjük az adatok vizualizálásának alapjait. Először egyszerű alapbeállítások mellett egy histogram típusú vizualizációt készítünk.

ggplot(  
 data = gapminder,   
 mapping = aes(x = gdpPercap)  
) +   
 geom\_histogram()



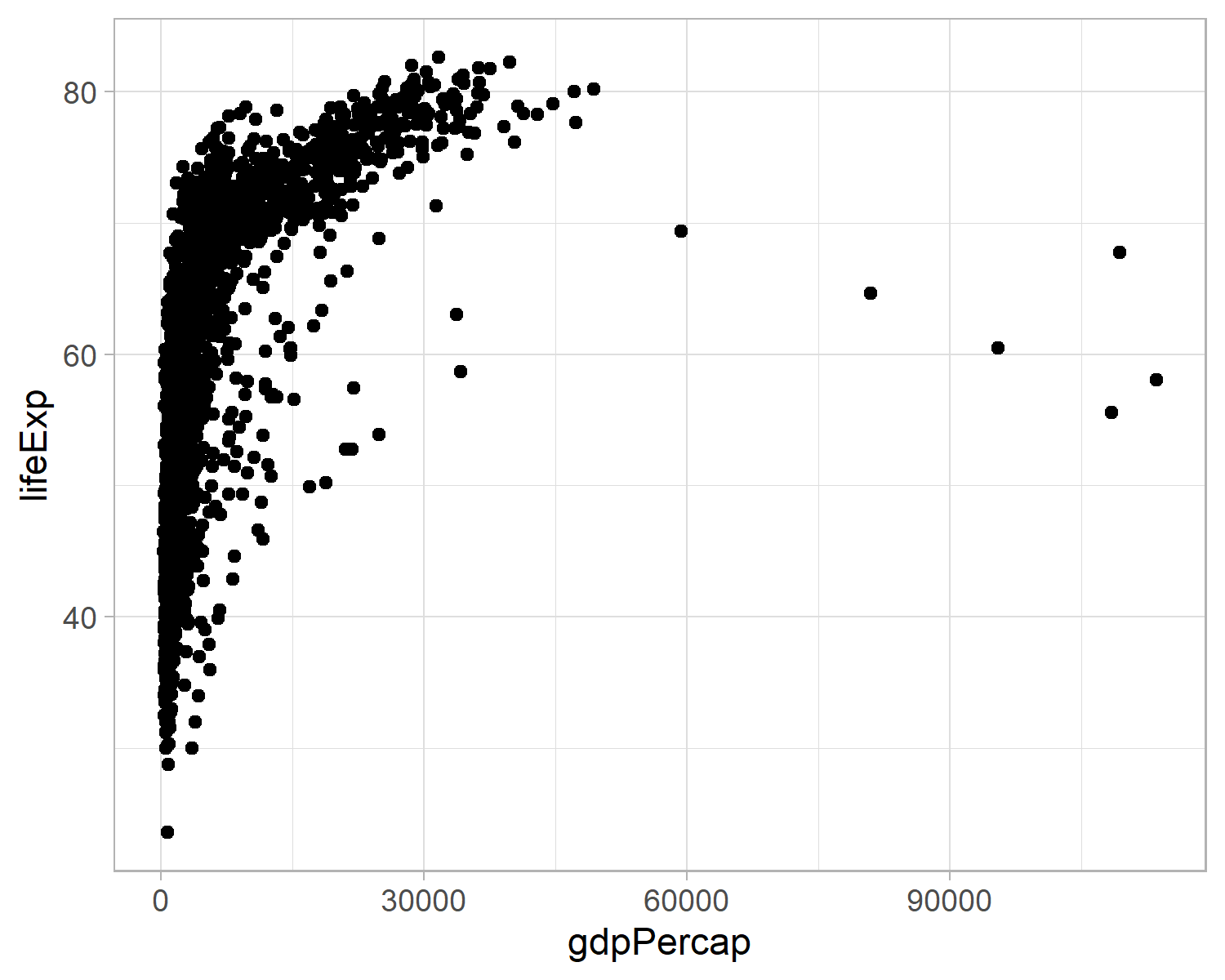
Lehetőségünk van arra, hogy az alakzat színét megváltoztassuk. A használható színek és színkódok megtalálhatóak a ggplot2 leírásában: <https://ggplot2-book.org/scale-colour.html>

ggplot(  
 data = gapminder,  
 mapping = aes(x = gdpPercap)  
) +  
 geom\_histogram(fill = "yellow", colour = "green")



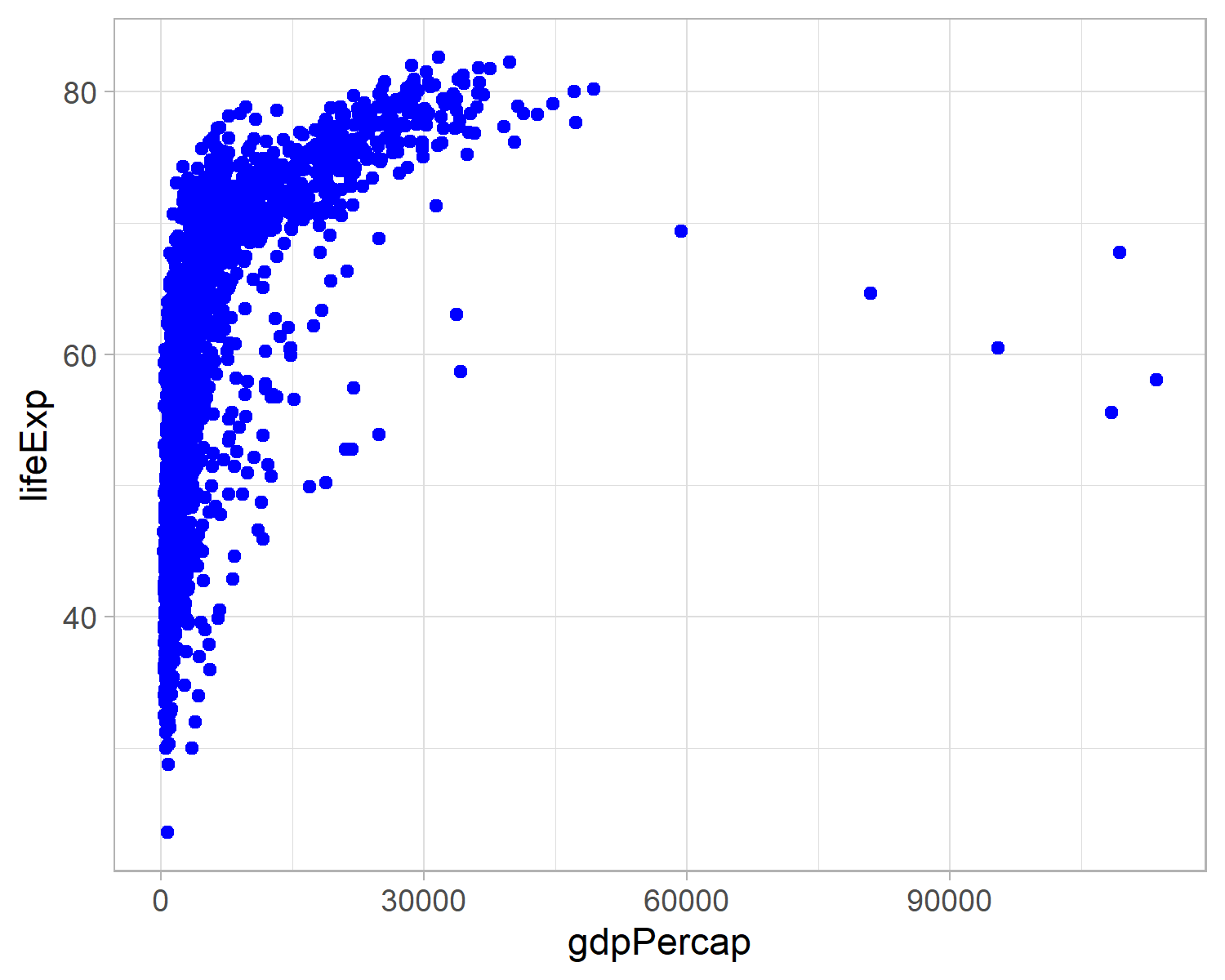
Meghatározhatjuk külön-külön a histogram x és y tengelyén ábrázolni kívánt adatokat és választhatjuk azok pontszerű ábrázolását is.

ggplot(  
 data = gapminder,  
 mapping = aes(  
 x = gdpPercap,  
 y = lifeExp  
 )  
) +  
 geom\_point()



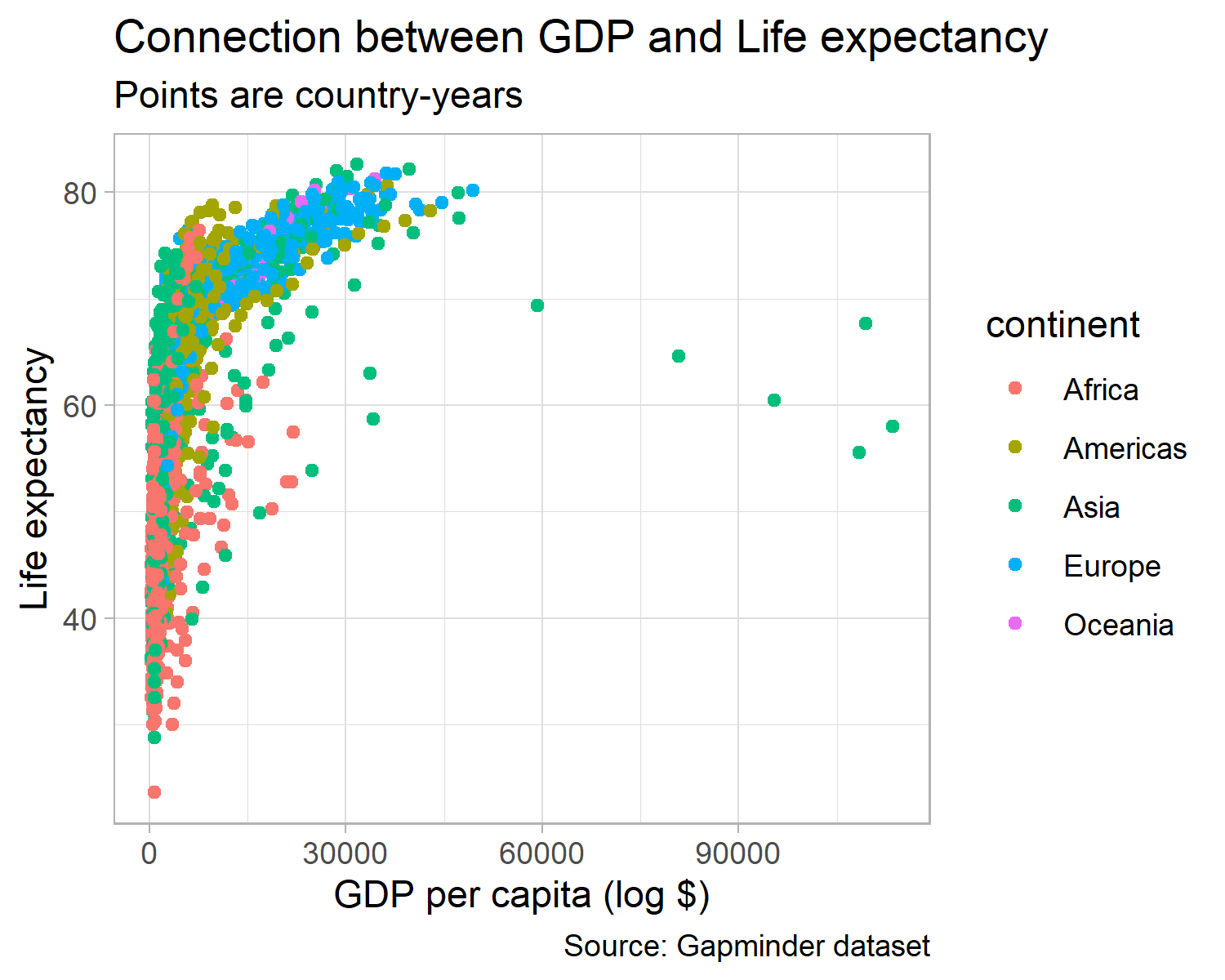
Ahogy az előzőekben, itt is megváltoztathatjuk az ábra színét.

ggplot(  
 data = gapminder,  
 mapping = aes(  
 x = gdpPercap,  
 y = lifeExp  
 )  
) +  
 geom\_point(colour = "blue")



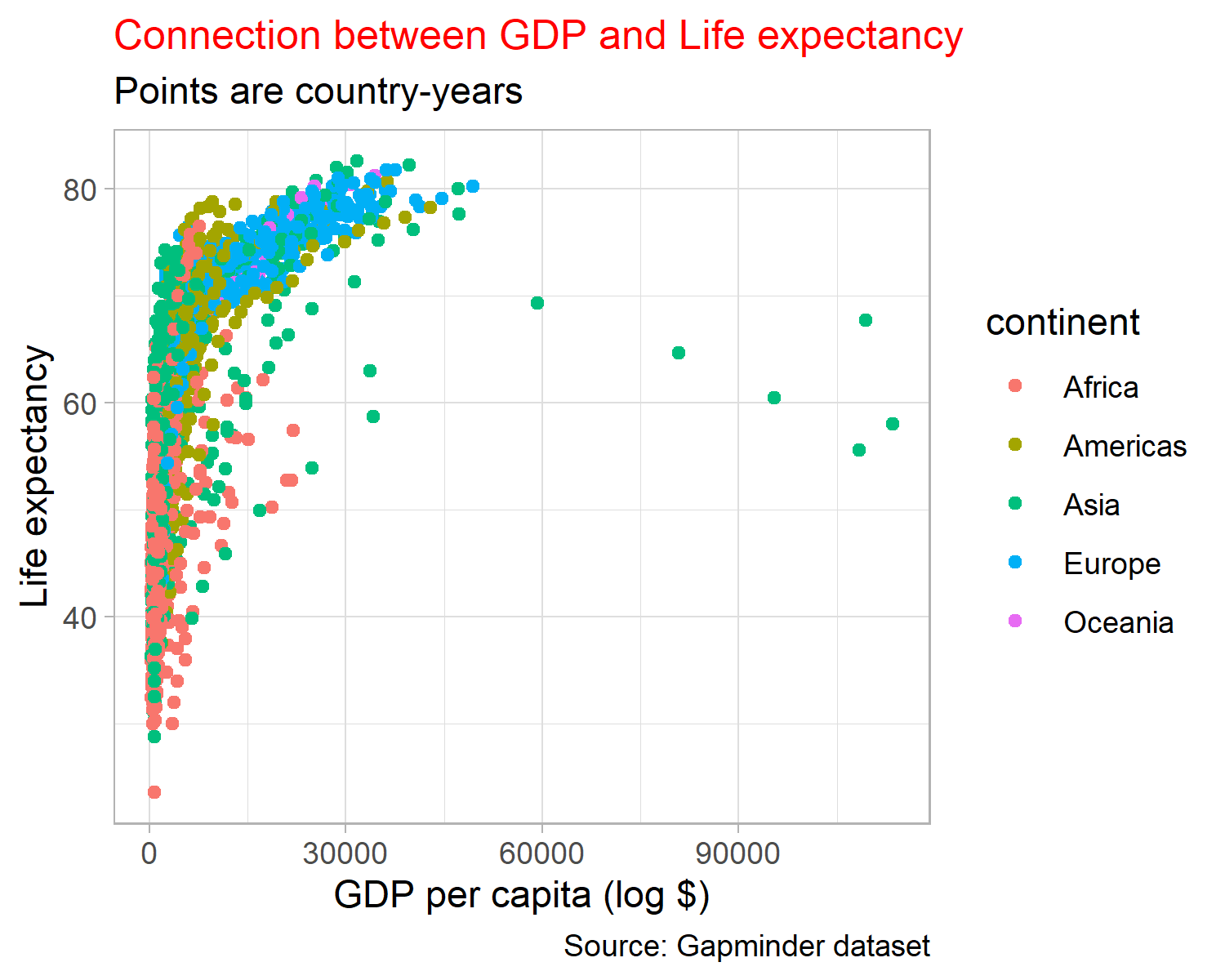
Az fenti script kibővítésével az egyes kontinensek adatait különböző színnel ábrázolhatjuk, az x és y tengelyt elnevezhetjük, a histogramnak címet és alcímet adhatunk, illetve az adataink forrását is feltüntethetjük az alábbi módon:

ggplot(  
 data = gapminder,  
 mapping = aes(  
 x = gdpPercap,  
 y = lifeExp,  
 color = continent  
 )  
) +   
 geom\_point() +  
 labs(  
 x = "GDP per capita (log $)",   
 y = "Life expectancy",  
 title = "Connection between GDP and Life expectancy",  
 subtitle = "Points are country-years",  
 caption = "Source: Gapminder dataset"  
 )



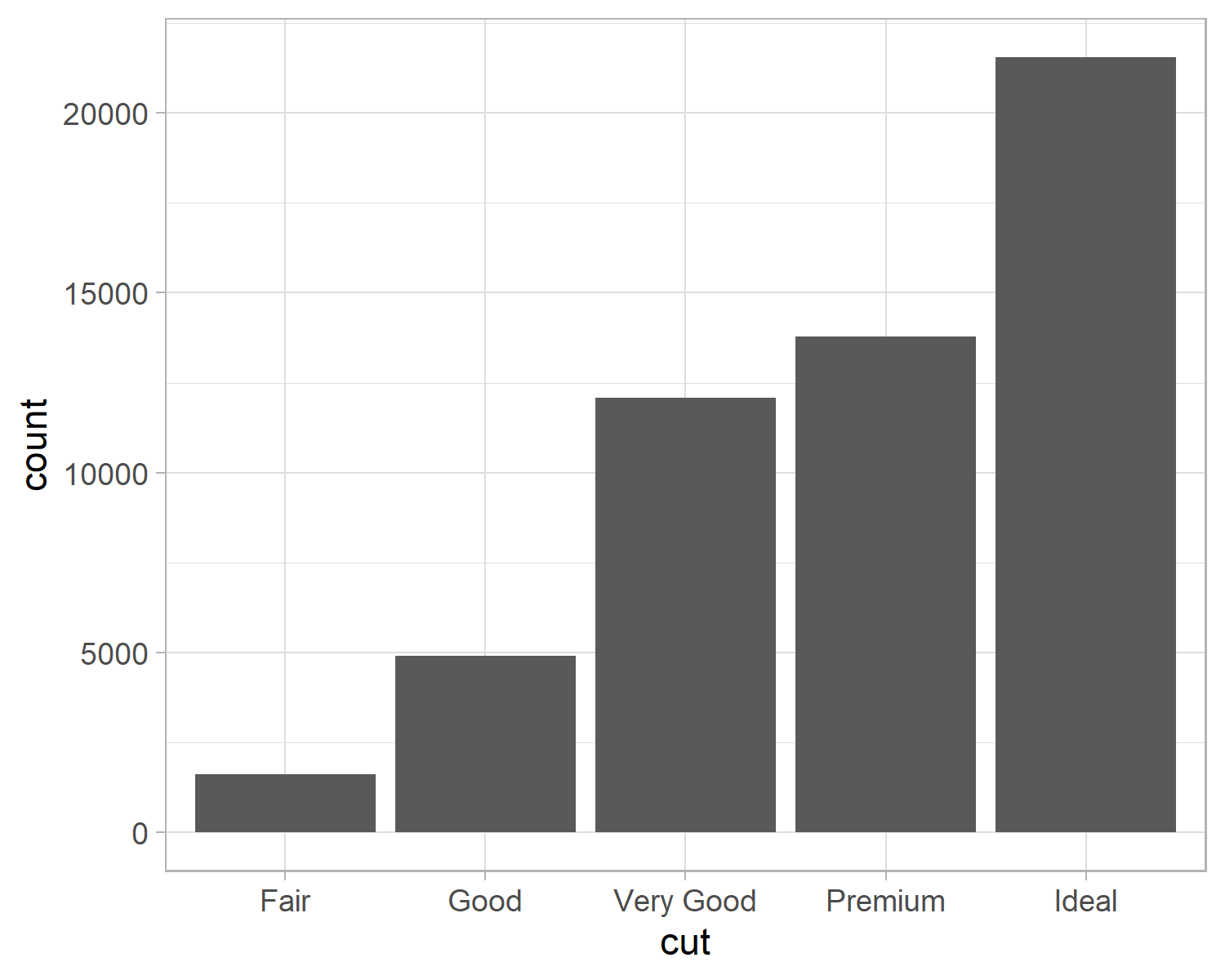
Az ábrán található feliratok méretének, betűtípusának és betűszínének megválasztásra is lehetőségünk van.

ggplot(  
 data = gapminder,  
 mapping = aes(  
 x = gdpPercap,  
 y = lifeExp,  
 color = continent  
 )  
) +   
 geom\_point() +  
 labs(  
 x = "GDP per capita (log $)",   
 y = "Life expectancy",  
 title = "Connection between GDP and Life expectancy",  
 subtitle = "Points are country-years",  
 caption = "Source: Gapminder dataset"  
 ) +  
 theme(plot.title = element\_text(  
 size = 12,   
 colour = "red"  
 ))



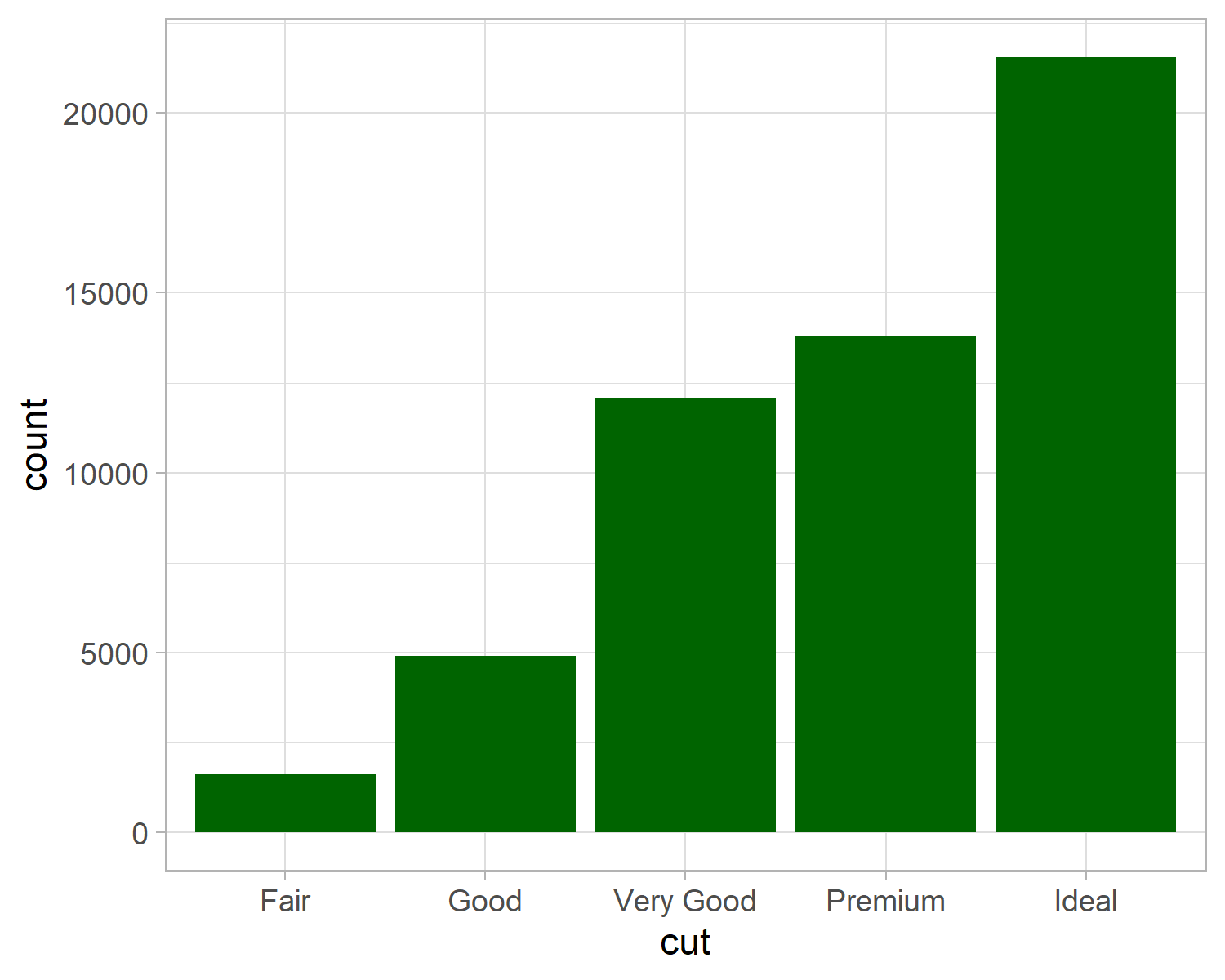
Készíthetünk oszlopdiagramot is, amit a ggplot2 diamonds adatkészletén személtetünk

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x = cut))



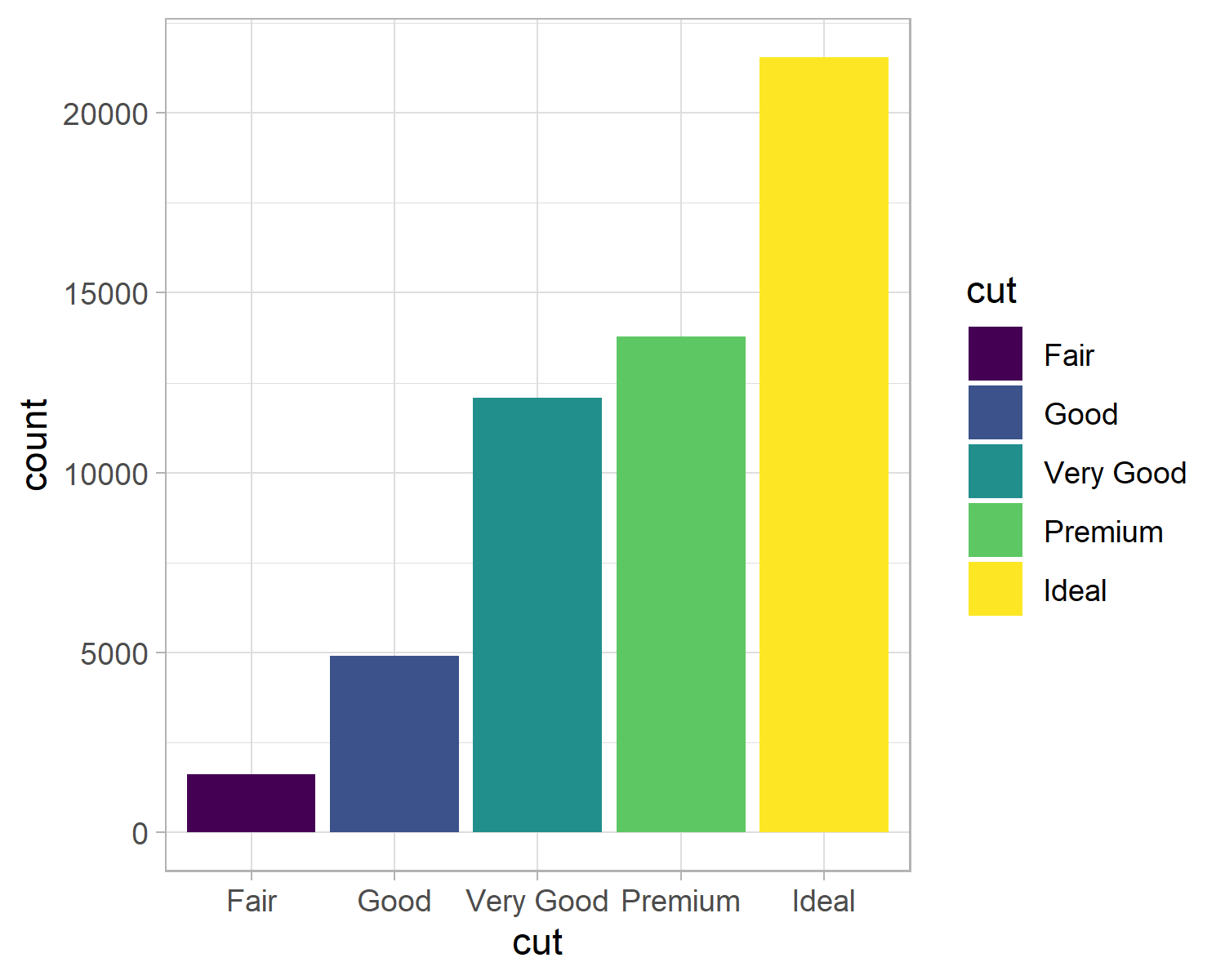
Itt is lehetőségünk van arra, hogy a diagram színét megváltoztassuk.

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x = cut), fill = "darkgreen")

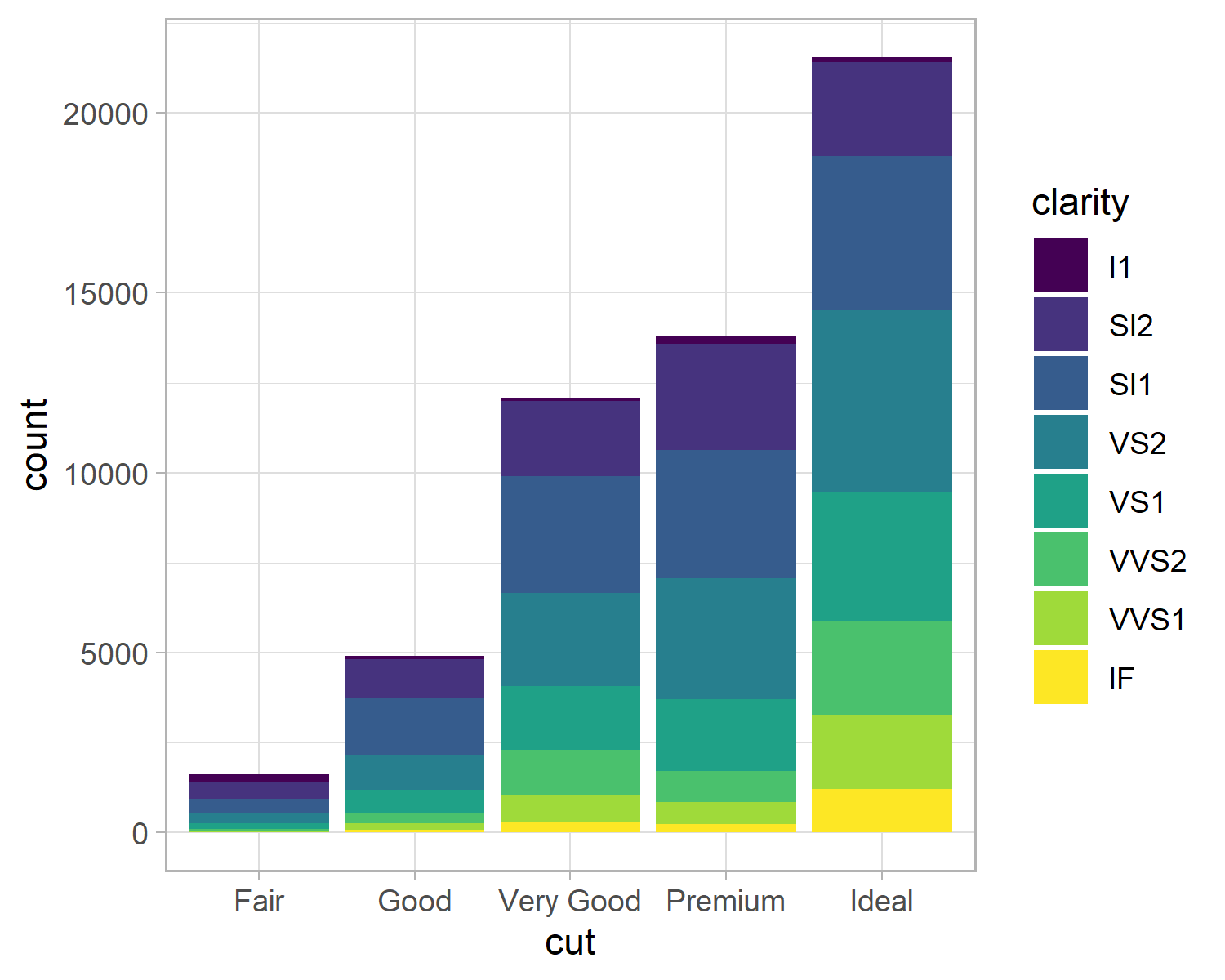


De arra is lehetőségünk van, hogy az egyes oszlopok eltérő színűek legyenek.

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x = cut, fill = cut))

 Arra is van lehetőségünk, hogy egyszerre több változót is ábrázoljunk.

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x = cut, fill = clarity))



Arra ggplot2 segítségével arra is lehetőségünk van, hogy csv-ből beolvasott adatainkat vizualizáljuk.

plot\_cap\_1 <- read.csv("data/plot\_cap\_1.csv", head = TRUE, sep = ";")   
ggplot(plot\_cap\_1, aes(Year, fill = Subtopic)) +   
 scale\_x\_discrete(limits = c(1957, 1958, 1959, 1960, 1961, 1962, 1963)) +  
 geom\_bar(position = "dodge") +   
 labs(  
 x = NULL, y = NULL,   
 title = "A Magyar Közlönyben kihirdetett agrárpolitikai jogszabályok",   
 subtitle = "N=445"  
 ) +   
 coord\_flip() + # az ábra tipusa  
 theme\_minimal() +  
 theme(plot.title = element\_text(size = 12))

A csv-ből belolvasott adatainkból kördiagramot is készíthetünk

pie <- read.csv("data/pie.csv", head = TRUE, sep = ";")  
  
ggplot(pie, aes(x = "", y = value, fill = Type)) +  
 geom\_bar(stat = "identity", width = 1) +  
 coord\_polar("y", start = 0) +  
 scale\_fill\_brewer(palette = "GnBu") +  
 labs(  
 title = "A Magyar Közlönyben megjelent jogszabályok típusai",  
 subtitle = "N = 445"  
 ) +  
 theme\_void()

Arun, Rajkumar, Venkatasubramaniyan Suresh, CE Veni Madhavan, and MN Narasimha Murthy. 2010. “On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations.” In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 391–402. Springer.

Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani. 2010. “Sentiwordnet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.” In *Lrec*, 10:2200–2204.

Bar, Daniel, Torsten Zesch, and Iryna Gurevych. 2011. “A Reflective View on Text Similarity.” *Proceedings of Recent Advances in Natural Language Processing*, 515–20.

Blei, David M, Andrew Y Ng, and Michael I Jordan. 2003. “Latent Dirichlet Allocation.” *Journal of Machine Learning Research* 3 (Jan): 993–1022.

Brady, Henry E. 2019. “The Challenge of Big Data and Data Science.” *Annual Review of Political Science* 22 (1): 297–323. <https://doi.org/10.1146/annurev-polisci-090216-023229>.

Burtejin, Zorgit. 2016. “Csoportosítás (Klaszterezés).” In *Kvantitatív Szövegelemzés és Szövegbányászat a Politikatudományban*, edited by Miklós Sebők, 85–101. Budapest: L’Harmattan.

Cao, Juan, Tian Xia, Jintao Li, Yongdong Zhang, and Sheng Tang. 2009. “A Density-Based Method for Adaptive LDA Model Selection.” *Neurocomputing* 72 (7-9): 1775–81.

Deveaud, Romain, Eric SanJuan, and Patrice Bellot. 2014. “Accurate and Effective Latent Concept Modeling for Ad Hoc Information Retrieval.” *Document Numérique* 17 (1): 61–84.

Griffiths, T. L., and M. Steyvers. 2004. “Finding Scientific Topics.” *Proceedings of the National Academy of Sciences* 101 (Supplement 1): 5228–35. <https://doi.org/10.1073/pnas.0307752101>.

Grimmer, Justin, and Brandon M Stewart. 2013. “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.” *Political Analysis* 21 (3): 267–97.

Hjorth, Frederik, Robert Klemmensen, Sara Hobolt, Martin Ejnar Hansen, and Peter Kurrild-Klitgaard. 2015. “Computers, Coders, and Voters: Comparing Automated Methods for Estimating Party Positions.” *Research & Politics* 2 (2): 2053168015580476.

Jacobi, Carina, Wouter Van Atteveldt, and Kasper Welbers. 2016. “Quantitative Analysis of Large Amounts of Journalistic Texts Using Topic Modelling.” *Digital Journalism* 4 (1): 89–106.

Kwartler, Ted. 2017. *Text Mining in Practice with R*. John Wiley & Sons.

Ladd, John R. 2020. “Understanding and Using Common Similarity Measures for Text Analysis.” *The Programming Historian* 9.

Laver, Michael, Kenneth Benoit, and John Garry. 2003. “Extracting Policy Positions from Political Texts Using Words as Data.” *American Political Science Review*, 311–31.

Laver, Michael, and John Garry. 2000. “Estimating Policy Positions from Political Texts.” *American Journal of Political Science*, 619–34.

Liu, Bing. 2010. “Sentiment Analysis and Subjectivity.” *Handbook of Natural Language Processing* 2 (2010): 627–66.

Loughran, Tim, and Bill McDonald. 2011. “When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks.” *The Journal of Finance* 66 (1): 35–65.

Máté, Ákos, Miklós Sebők, and Tamás Barczikay. 2021. “The Effect of Central Bank Communication on Sovereign Bond Yields: The Case of Hungary.” Edited by Hiranya K. Nath. *PLOS ONE* 16 (2): e0245515. <https://doi.org/10.1371/journal.pone.0245515>.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” *arXiv Preprint arXiv:1301.3781*. <http://arxiv.org/abs/1301.3781>.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. “Advances in Pre-Training Distributed Word Representations.” In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. “Glove: Global Vectors for Word Representation.” In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–43.

Phan, Xuan-Hieu, Le-Minh Nguyen, and Susumu Horiguchi. 2008. “Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-Scale Data Collections.” In *Proceedings of the 17th International Conference on World Wide Web*, 91–100.

Roberts, Margaret E, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G Rand. 2014. “Structural Topic Models for Open-Ended Survey Responses.” *American Journal of Political Science* 58 (4): 1064–82.

Russel, Stuart, and Peter Norvig. 2005. *Mesterséges Intelligencia*. Panem Kft.

Schütze, Hinrich, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*. Vol. 39. Cambridge University Press Cambridge.

Sieg, Adrien. 2018. “Text Similarities : Estimate the Degree of Similarity Between Two Texts.” *Medium*.

Silge, Julia, and David Robinson. 2017. *Text Mining with R: A Tidy Approach*. " O’Reilly Media, Inc.".

Slapin, Jonathan B, and Sven-Oliver Proksch. 2008. “A Scaling Model for Estimating Time-Series Party Positions from Texts.” *American Journal of Political Science* 52 (3): 705–22.

Spirling, Arthur, and Pedro L Rodriguez. Forthcoming. “Word Embeddings.” *Journal of Politics*, Forthcoming.

Straka, Milan, and Jana Straková. 2017. “Tokenizing, Pos Tagging, Lemmatizing and Parsing Ud 2.0 with Udpipe.” In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 88–99.

Szarvas, György, Richárd Farkas, and András Kocsor. 2006. “A Multilingual Named Entity Recognition System Using Boosting and C4. 5 Decision Tree Learning Algorithms.” In *International Conference on Discovery Science*, 267–78. Springer.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. 2011. *Bevezetés Az Adatbányászatba*. Panem Kft.

Tikk, Domonkos. 2007. *Szövegbányászat*. Budapest: Typotext.

Üveges, István. 2019. “Named Entity Recognition in the Miskolc Legal Corpus.”

Vincze, Veronika. 2019. “Beszéd-és Nyelvelemző Szoftverek.” In *Beszéd- és Nyelvelemző Szoftverek a Versenyképességért és Az Esélyegyenlőségért HunCLARIN Korpuszok és Nyelvtechnológiai Eszközök a bölcsészet- és társadalomtudományokban*, 7–22. Szeged.

Wang, Jiapeng, and Yihong Dong. 2020. “Measurement of Text Similarity: A Survey.” *Information* 11 (9): 421. <https://doi.org/10.3390/info11090421>.

Welbers, Kasper, Wouter Van Atteveldt, and Kenneth Benoit. 2017. “Text Analysis in R.” *Communication Methods and Measures* 11 (4): 245–65.

Wickham, Hadley, and Garrett Grolemund. 2016. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. " O’Reilly Media, Inc.".

Young, Lori, and Stuart Soroka. 2012. “Affective News: The Automated Coding of Sentiment in Political Texts.” *Political Communication* 29 (2): 205–31.

Zsibrita, János, Veronika Vincze, and Richárd Farkas. 2013. “Magyarlanc: A Tool for Morphological and Dependency Parsing of Hungarian.” In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, 763–71.

1. Az R Windows, OS X és Linux változatai itt érhetőek el: <https://cloud.r-project.org/>. Az RStudio pedig innen érhető el: <https://www.rstudio.com/products/rstudio/download/>. [↑](#footnote-ref-23)
2. A quanteda.textplots csomag több megoldást is kínál az ábrák elkészítésére. Mivel ezek a megoldások kifejezetten a quanteda elemzések ábrázolására készültek, ezért rövid egysoros függvényekkel tudunk gyorsan ábrákat készíteni. A hátrányuk, hogy kevésbé tudjuk “személyre szabni” az ábráinkat, mint a ggplot2 példák esetében. A quanteda.textplots megoldásokat ezen a linken demonstrálják a csomag készítői: <https://quanteda.io/articles/pkgdown/examples/plotting.html> [↑](#footnote-ref-26)
3. Vegyük észre azt is, hogy több olyan kifejezés van, melyek csak ragozásukban térnek el: Unió, Unióhoz, tagja, tagjának. Ezeket a kifejezéseket a kutatói szándék¬ függvényében azonos alakúra hozhatjuk, hogy egy egységként jelenjenek meg. – Az elemzések többségében a szövegelőkészítés egyik kiinduló lépése a szótövesítés vagy a lemmatizálás, előbbi a szavak toldalékainak levágását jelöli, utóbbi a szavak szótári alakra való visszaalakítását. A ragozás eltávolítását illetően elöljáróban annyit érdemes megjegyezni, hogy az agglutináló, vagyis ragasztó nyelvek esetén, mint amilyen a magyar is, a toldalékok eltávolítása gyakran igen komoly kihívást jelent. Nem csak a toldalékok formája lehet igen sokféle, de az is előfordulhat, hogy a tőszó nem egyezik meg a toldalék levágásával keletkező szótővel. ilyen például a vödröt kifejezés, melynek szótöve a „vödr“, de a nyelvtanilag helyes tőszó a vödör, hasonlóan a majmok kifejezés esetén a szótő a „majm” lesz, míg a nyelvtanilag helyes tőszó a majom. Emiatt a toldalékok levágását a magyar nyelvű szövegek esetén megfelelő körültekintéssel kell végezni. [↑](#footnote-ref-35)
4. Az .rda több objektum egyetlen fájlba történő mentésére alkalmas, mentése a save(), betöltése a load() paranccsal lehetséges. Az .rds egyetlen objekumot ment el a saveRDS() paranccsal, beolvasni pedig readRDS() paranccsal lehet. [↑](#footnote-ref-41)
5. A *tidyverse* megközelítés miatt a kötetben szereplő R kód követi a “The tidyverse style guide” dokumentációt (<https://style.tidyverse.org/>) [↑](#footnote-ref-43)
6. Az RStudio-ban a pipe operátor billentyű kombinációja a Ctrl + Shift + M [↑](#footnote-ref-45)
7. Köszönjük Andrew Heissnek a kitűnő példát. [↑](#footnote-ref-46)
8. A könyv terjedelme miatt ezt a témát itt csak bemutatni tudjuk, de minden részletre kiterjedően nem tudunk elmélyülni benne. A témában nagyon jól használható online anyagok találhatóak az RStudio GitHub tárhelyén (<https://github.com/rstudio/cheatsheets/raw/master/strings.pdf>), illetve [Wickham and Grolemund](#ref-wickham2016r) ([2016](#ref-wickham2016r)) 14. fejezetében. [↑](#footnote-ref-53)
9. Az itt használt kódok az alábbiakon alapulnak: <https://rdrr.io/cran/quanteda/man/dfm_weight.html>, <https://rdrr.io/cran/quanteda/man/dfm_tfidf.html> [↑](#footnote-ref-74)
10. A lubridate használatának részletes leírása megtalálható az alábbi linken: <https://rawgit.com/rstudio/cheatsheets/master/lubridate.pdf> [↑](#footnote-ref-81)
11. A lambda leírása megtalálható: <https://quanteda.io/reference/textstat_collocations.html> [↑](#footnote-ref-83)
12. A különböző indexek leírása megtalálható az alábbi linken: <https://quanteda.io/reference/textstat_lexdiv.html> [↑](#footnote-ref-86)
13. ([Schütze, Manning, and Raghavan 2008](#Xa269f770aa4a47943f7101d538748afa012fcd7)) [↑](#footnote-ref-91)
14. Bővebben lásd például: ([Liu 2010](#ref-liuSentimentAnalysisSubjectivity2010)) [↑](#footnote-ref-98)
15. A lehetséges, területspecifikus szótáralkotási módszerekről részletesebben ezekben a cikkekben lehet olvasni: [Laver and Garry](#ref-laver2000estimating) ([2000](#ref-laver2000estimating)); [Young and Soroka](#ref-young2012affective) ([2012](#ref-young2012affective)); [Loughran and McDonald](#ref-loughranWhenLiabilityNot2011) ([2011](#ref-loughranWhenLiabilityNot2011)); [Máté, Sebők, and Barczikay](#ref-mateEffectCentralBank2021) ([2021](#ref-mateEffectCentralBank2021)) [↑](#footnote-ref-99)
16. A szótár és dokumentációja elérhető az alábbi linken: <https://github.com/aesuli/SentiWordNet> [↑](#footnote-ref-100)
17. A quanteda.dictionaries csomag leírása és a benne található szótárak az alábbi github linken érhetőek el: <https://github.com/kbenoit/quanteda.dictionaries> [↑](#footnote-ref-102)
18. <http://www2.imm.dtu.dk/pubdb/pubs/6010-full.html> [↑](#footnote-ref-104)
19. <https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html> [↑](#footnote-ref-106)
20. <http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm> [↑](#footnote-ref-108)
21. A szentiment elemzéshez gyakran használt csomag még a tidytext. Az online is szabadon elérhető [Silge and Robinson](#ref-silge2017text) ([2017](#ref-silge2017text)) 2. fejezetében részletesen is bemutatják a szerzők a tidytext munkafolyamatot (<https://www.tidytextmining.com/sentiment.html>). [↑](#footnote-ref-110)
22. A korpusz a Hungarian Comapartive Agendas Project keretében készült és regisztáció után, kutatási célra elérhető az alábbi linken: <https://cap.tk.hu/a-media-es-a-kozvelemeny-napirendje>. [↑](#footnote-ref-112)
23. A korpusz, a szótár és az elemzés teljes dokumentációja elérhető az alábbi github linken: <https://github.com/poltextlab/central_bank_communication>, a teljes elemzés ([Máté, Sebők, and Barczikay 2021](#ref-mateEffectCentralBank2021))elérhető: <https://doi.org/10.1371/journal.pone.0245515> [↑](#footnote-ref-114)
24. <https://milab.tk.hu/hu> A szótár és a hozzátartozó dokumentáció elérhető az alábbi github oldalon: <https://github.com/poltextlab/sentiment_hun> [↑](#footnote-ref-118)
25. A csoportosított adatokkal való munka bővebb bemutatását lsd. a Függelékben [↑](#footnote-ref-121)
26. A témával részletesebben is foglalkoztunk a [Máté, Sebők, and Barczikay](#ref-mateEffectCentralBank2021) ([2021](#ref-mateEffectCentralBank2021)) tanulmányban, ahol egy saját monetáris szentiment szótárt mutatunk be. Az implementáció és a hozzá tartozó R forráskód a nyilvános <https://doi.org/10.6084/m9.figshare.13526156.v1> linken. [↑](#footnote-ref-124)
27. a kód részben az alábbiakon alapul: tidytextmining.com/topicmodeling.html Az általunk is használt topicmodels csomag interfészt biztosít az LDA modellek és a korrelált témamodellek (CTM) C kódjához, valamint az LDA modellek illesztéséhez szükséges C ++ kódhoz. [↑](#footnote-ref-132)
28. A teljes törvényeket és a meta adatokat tartalmazó adatbázisokat a <https://cap.tk.hu/> honlapról lehet letölteni. [↑](#footnote-ref-135)
29. <http://brooksandrew.github.io/simpleblog/articles/latent-dirichlet-allocation-under-the-hood/> [↑](#footnote-ref-139)
30. A ldatuning R csomagban további indikátor implementációja található, ami a perplexityhez hasonlóan minimalizásra alapoz ([Arun et al. 2010](#ref-arun2010finding); [Cao et al. 2009](#ref-cao2009density)), illetve maximalizálásra ([Deveaud, SanJuan, and Bellot 2014](#ref-deveaud2014accurate); [Griffiths and Steyvers 2004](#ref-griffithsFindingScientificTopics2004)) [↑](#footnote-ref-142)
31. a tidy formátumról bővebben: <https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html> [↑](#footnote-ref-143)
32. A kovariancia megadja két egymástól különböző változó együttmozgását. Kis értékei gyenge, nagy értékei erős lineáris összefüggésre utalnak. [↑](#footnote-ref-152)
33. <https://cap.tk.hu/hu> [↑](#footnote-ref-153)
34. Az stm csomaghoz tartozó leírás: <https://cran.r-project.org/web/packages/stm/vignettes/stmVignette.pdf> [↑](#footnote-ref-155)
35. Egy kiváló tanulmányban [Spirling and Rodriguez](#ref-spirlingword) ([Forthcoming](#ref-spirlingword)) összehasonlítják a Word2Vec és GloVe módszereket, különböző paraméterekkel, adatbázisokkal. Amennyiben valakit komolyabban érdekelnek a szóbeágyazás gyakorlati alkalmazásának a részletei annak mindenképp ajánljuk elolvasásra. [↑](#footnote-ref-161)
36. A Magyar CAP Projekt által kezelt adatbázisok itt megtalálhatóak: <https://cap.tk.hu/adatbazisok> [↑](#footnote-ref-162)
37. A futtatásra használt PC konfiguráció: CPU: Intel Core i5-4460 (3.2GHz); RAM: 16GB [↑](#footnote-ref-164)
38. Az egyik legelterjedtebb dimenzionalitás csökkentő eljárás a főkomponens-analízis (principal component analysis), illetve szintén gyakran használt az irodalomban az úgynevezett t-SNE (t-distributed stochastic neighbor embedding). [↑](#footnote-ref-165)
39. A quanteda.textplots csomag több megoldást is kínál az ábrák elkészítésére. Mivel ezek a megoldások kifejezetten a quanteda elemzések ábrázolására készültek, ezért rövid egysoros függvényekkel tudunk gyorsan ábrákat készíteni. A hátrányuk, hogy kevésbé tudjuk “személyre szabni” az ábráinkat, mint a ggplot2 példák esetében. A quanteda.textplots megoldásokat ezen a linken demonstrálják a csomag készítői: <https://quanteda.io/articles/pkgdown/examples/plotting.html> [↑](#footnote-ref-170)
40. Azért nem a Vona Gábor beszédét választottuk, mert az gyaníthatóan egy kiugró érték ami nem reprezentálja a sokaságot megfelelően. [↑](#footnote-ref-175)
41. Amatrix\_raster\_plot() függvényt a nandb csomag feltelepítése után tudjuk csak használni. [↑](#footnote-ref-187)
42. Az UDPipe elérhetősége: <http://lindat.mff.cuni.cz/services/udpipe> [↑](#footnote-ref-193)
43. <https://analyticsindiamag.com/top-10-r-packages-for-natural-language-processing-nlp/> [↑](#footnote-ref-202)
44. <https://spacyr.quanteda.io/articles/using_spacyr.html> [↑](#footnote-ref-204)
45. Részletes leírása: <https://spacy.io/usage/linguistic-features#named-entities> [↑](#footnote-ref-206)
46. <https://towardsdatascience.com/extend-named-entity-recogniser-ner-to-label-new-entities-with-spacy-339ee5979044> [↑](#footnote-ref-208)
47. <https://tutorials.quanteda.io/basic-operations/fcm/fcm/> [↑](#footnote-ref-210)