Szövegbányászat és mesterséges intelligencia R-ben

Sebők Miklós, Ring Orsolya

2021-04-12 15:37:39

# 1 Bevezetés

Jelen kötet a Kvantitatív szövegelemzés és szövegbányászat a politikatudományban (L’Harmattan, 2016) című könyv folytatásaként és egyben kiegészítéseként a szövegbányászat és a mesterséges intelligencia társadalomtudományi alkalmazásának gyakorlatába nyújt bevezetést. 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.

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. A kézirat a magyar tankönyvpiacon az elsőnek számít a tekintetben, hogy a társadalomtudományban használatos kvantitatív szövegelemzési eljárásokat részletesen, lépésről-lépésre ismerteti, kezdve a megfelelő korpusz kialakításához szükséges ismeretektől, a különböző szövegbányászati módszerek (szózsák, dokumentum-kifejezés mátrix, a névelem-felismerés, az osztályozás, illetve a csoportosítás feladataira), illetve az egyszerűbb szövegösszehasonlítási-feladatok áttekintésén át, egészen a felügyelt és felügyelet nélküli gépi tanulásig, a politikatudományi vizsgálatok során leggyakrabban használatos R programnyelven készült programok segítségével.

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 végigkísérő konkrét példák segítségével pedig akár reprodukálhatja is ezen eljárásokat saját kutatásában. Mindezt a kötet függelékében helyet kapó R-scriptek részletes leírása is segíti majd. A kötet két fő célcsoportjaként így a társadalomtudományi kutatói és felsőoktatási közösséget határozzuk meg, valamint rögzítjük, hogy a kvantitatív szövegelemzés területén belül elsődlegesen a dokumentum- és tartalomelemzési módszertanhoz kapcsolódunk.

A könyvben ugyancsak helyet kap a fontosabb fogalmak magyar és angol nyelvű szószedete, valamint a további olvasásra ajánlott szakirodalom szerepeltetése. Az oktatásban való közvetlen alkalmazást segíthetik továbbá a fejezetek végén megadott vizsgakérdések, illetve a kötet honlapján (qta.tk.mta.hu) szereplő további információk: gyakorlófeladatok (megoldásokkal), az egyes feladatokra alkalmazható scriptek és kereskedelmi programok bemutatása, a témával kapcsolatos prezentációk és további ajánlott irodalmak.

# 2 A kvantitatív szövegelemzés és szövegbányászat alapfogalmai

elso fejezet

# 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 package is megoldást kínál. Ezek közül a könyv írásakor a legnépszerűbbek a readr és a rio csomagok. A karakter kódolással a legjobban a tapasztalataink szerint a readr csomag read\_csv() megoldása bíkózik meg, ezért 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 fileban vannak (comma separated values), ahol az egyes értékeket vesszők (vagy egyéb speciális karakter) 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 fileokat, elég megadnunk egy működő elérési útvonalat.

**placeholder link, cserelni majd mukodore**

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 bírkózni. Az Excel fileokat 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 tudunk beolvasni egy adott mappából az összes .txt kiterjesztésű file-t, anélkül hogy bármilyen loop-ot kellene írnunk, vagy egyenként megadni a file-ok neveit. A \* karakter az azt jelenti ebben a környezetben, hogy bármilyen fájl, ami .txt-re végződik. Amennyiben a fájlok nevei tartalmaznak valamilyen meta adatot tartalmaznak, akkor ezt be tudjuk allítani 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 R rendelkezik saját, .Rds és .Rda kiterjesztésű, tömörített fájlformátummal. Mivel ezeket csak az R-ben nyithatjuk meg, érdemes a köztes, hosszadalmas számítást igénylő lépések elmentésére használni, a saveRDS() és a save() parancsokkal. Az openxlsx csomaggal .xls és .xlsx Excel formátumokba is tudunk exportálni, hogyha szükséges.

## 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.[[1]](#footnote-25) Vizuálisan a pipe operátor így néz ki: %>% és arra szolgál hogy a kódban több egymáshoz kapcsolódó műveletet egybefűzzűnk.[[2]](#footnote-27) 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.[[3]](#footnote-28) 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 használatára. A fejezetben bemutatott példák az alkalmazásoknak csak egy relatíve szűk körét mutatják be, ezért érdemes átolvasni a csomagokhoz tartozó dokumentációt, illetve ha van, akkor a működést demonstráló bemutató oldalakat is.

## 3.4 Muveletek a date framekkel

A data frame az egyik leghasznosabb és leggyakrabban használt adat tárolási mód az R-ben (a részletesebb leírás a [Függelékben](#data-frame) található) és ebben az alfejezetben azt mutatjuk be a dplyr és gapminder csomagok segíségével, hogy hogyan lehet hatékonyan dolgozni velük. 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 sorrendjet is megadja. A negatív kiválasztás is lehetséges, ebben az esetben egy - kell tennünk a nemkívánt 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

### 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 tehetjuk 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[[4]](#footnote-35)

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 elemzésünkhoz. 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 kotextusban 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 szövebányászati 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 csomomag 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)

Majd 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 fájlá 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 megkezdenünk. A szövegtisztításnél 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 “Szöveges dokumentumok importálása”[BE KELL MAJD ÍRNI A VÉLGLEGES FEJEZETSZÁMOT] című aljezetben 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 kopruszból. Ehhez először létrehozzuk a text1 nevű objektumot ami egy karaktervektoból 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 karekterláncok 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 elenevezésű 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.([**grimmer2013a?**](#ref-grimmer2013a))

##### 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 magyarláncnyelvi elemző segítségével, melyről a kötet függelékében, a Magyar nyelvű NLP és nyelvtechnológiai eszközök között szólunk részletesebben.

text1 <- "Példa az elokészítésre. Az elokészítést a szövetisztítással kell megkezdenünk. Az elokészített korpuszon elemzést végzünk"  
  
toks1 <- tokens(text1, remove\_punct = TRUE)  
  
elokészítés <- c("elokészítésre", "elokészítést", "elokészített")  
  
lemma <- rep("elokészítés", length(elokészítés))  
  
toks1 <- tokens\_replace(toks1, elokészítés, 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 alkaítjuk a szövegeinket számokká, ami lehetővé teszi, hogy utána különböző staisztikai műveleteket végezzünk velük.

A dokumentumk kifejezés mátrix minden sora egy dokumentum, minden oszlopa egy kifejezés, az oszlopokban szereplő változók pedig az egyes kifejezések számát mutatják meg az egyes dokumnetumokban. 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 sokumentum kifejezés mátrixot.

text <- c(  
 d1 = "Ez egy példa az elofeldolgozásra",  
 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 szövegbányáaszati 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 ([**Tikk2007:33-37?**](#ref-Tikk2007:33-37))

Az alábbiakban az 1999-es törvényszöveken szemlétetjük hogy egy 125 dokumentumból létrehozott mátrix segítségével milyen alapvető statisztikai műveleteket végezhetünk.[[5]](#footnote-51)

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 dokumnetum 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 dokumnetum 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 eltávolitjuk.

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. ([**tikk2007:63?**](#ref-tikk2007:63); [**russel2005a:742?**](#ref-russel2005a:742)) 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 ([**kwartler2017?**](#ref-kwartler2017); [**schütze2008?**](#ref-schütze2008) ; [**tikk2007a:63-81?**](#ref-tikk2007a:63-81))

## 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 mérte különböző eljárásokkal csökkenthető. Fontos tudni, hogy a dokumentumok vektortér reprezentációjában a szavak szövegen belüli sorrendjére és pozíciójára vonatkozó információ nem található meg. ([**russel2005b:742-744?**](#ref-russel2005b:742-744); [**kwartler2017:20?**](#ref-kwartler2017:20); [**welbers2017?**](#ref-welbers2017)) 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 mártixot 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 ([**tikk2007:33-37?**](#ref-tikk2007:33-37); [**grimmer2013?**](#ref-grimmer2013); [**welbers2017?**](#ref-welbers2017))

## 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 “Adarkezelés R-ben” fejezetben ismertetett mutate() függvény használazával két csoportra osztjuk a beszédeket. Ehhez először a string\_extract()segítségével meghatározzuk, hogy a kettéosztáshoz hasznáni 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 háturló 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.

Ezt követően 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[[6]](#footnote-59).

Majd a glimpse() függvény segítsével 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 alaovető satisztikai 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 beszédket is összehaonlíthatjuk 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 megkeresük a korábban már megismert head() függvény segítségével megnézhetünk közülük tetszőleges számút[[7]](#footnote-60).

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.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 ngarm-okat hozunk létre a tokenekből, majd kialakítjuk a dokumentumk kifejezés mátrixot (dfm) és elvégezzük a tf-idf szerinti súlyozást. A dfm\_tfidf()függvény kiszámolja a dokumentum gyakoriság inzverz 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) ngram-okról.

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 diverzitásat 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 különbözöségét, példánkban a CTTR indexet használjuk[[8]](#footnote-63).

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.

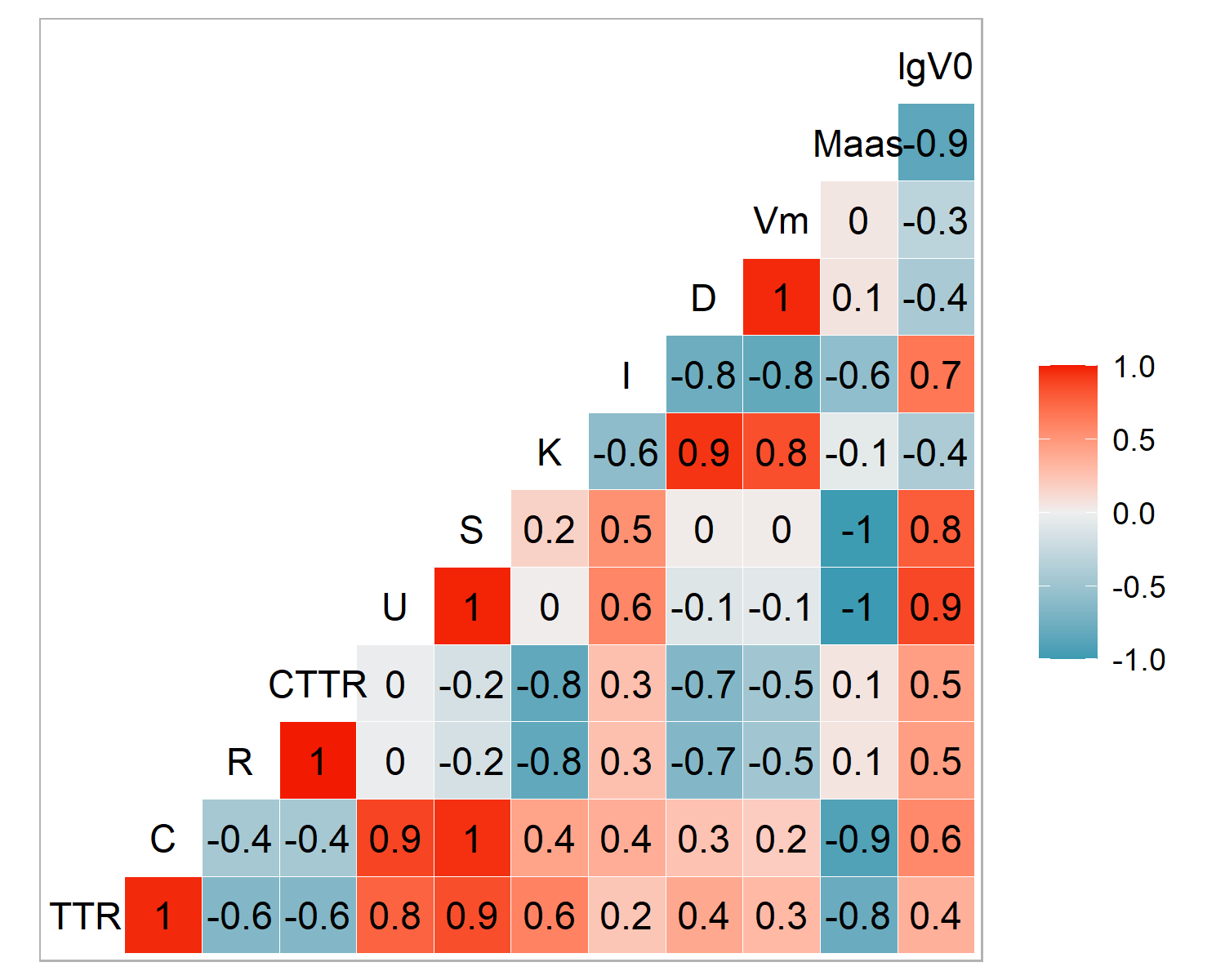
mineln\_dfm %>%  
 textstat\_lexdiv(measure = "all")  
#> document TTR C R CTTR U  
#> 1 antall\_jozsef\_1990 0.6465054 0.9490329 46.65601 32.99078 72.92298  
#> 2 bajnai\_gordon\_2009 0.7283044 0.9566410 28.18830 19.93214 73.23764  
#> 3 boross\_peter\_1993 0.7209677 0.9565427 31.09369 21.98656 75.23509  
#> 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 viszonyulanak 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 szövei 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()

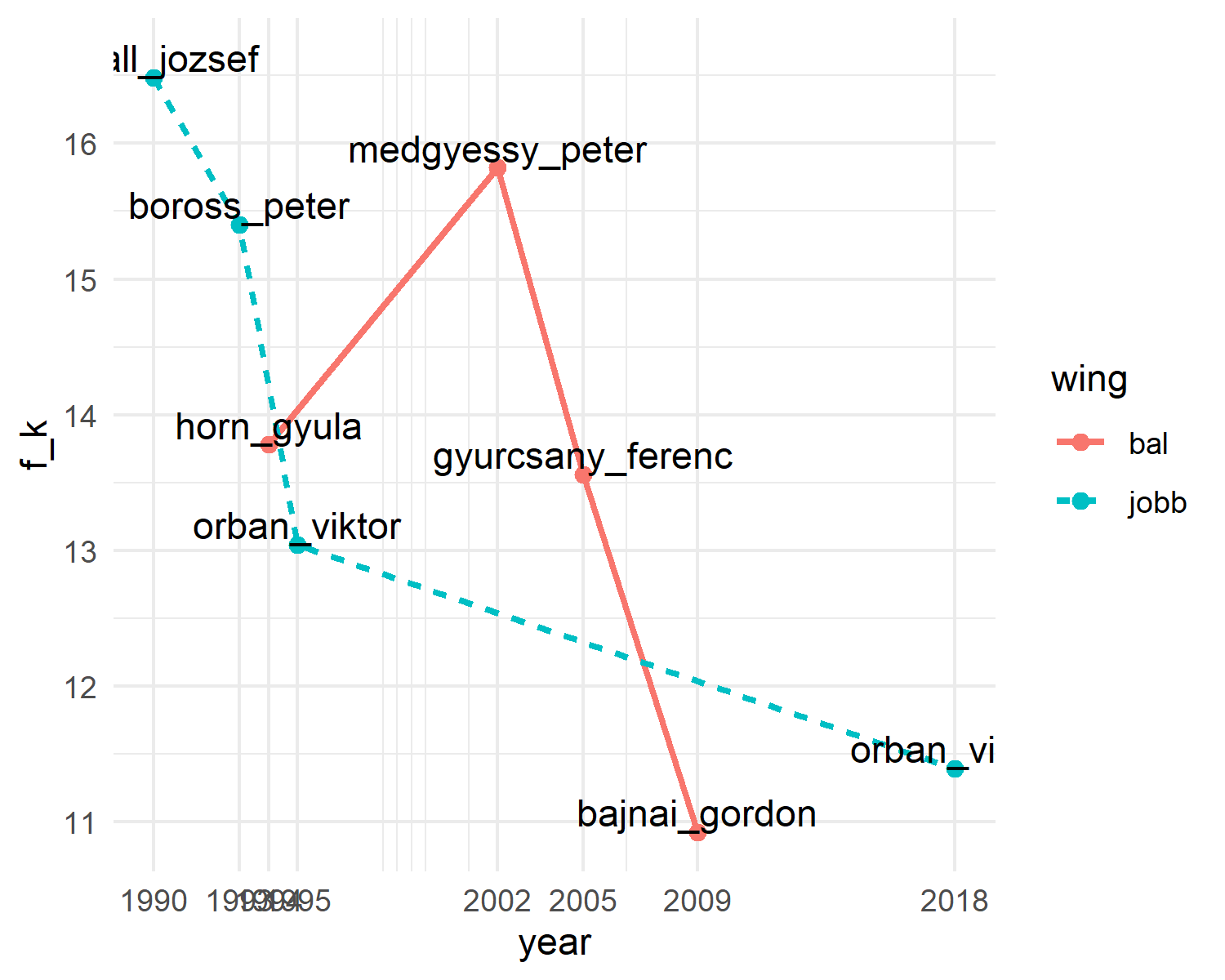


Figure 5.1: Az olvashatósági index alakulása

## 5.4 Összehasonlítás[[9]](#footnote-67)

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)))

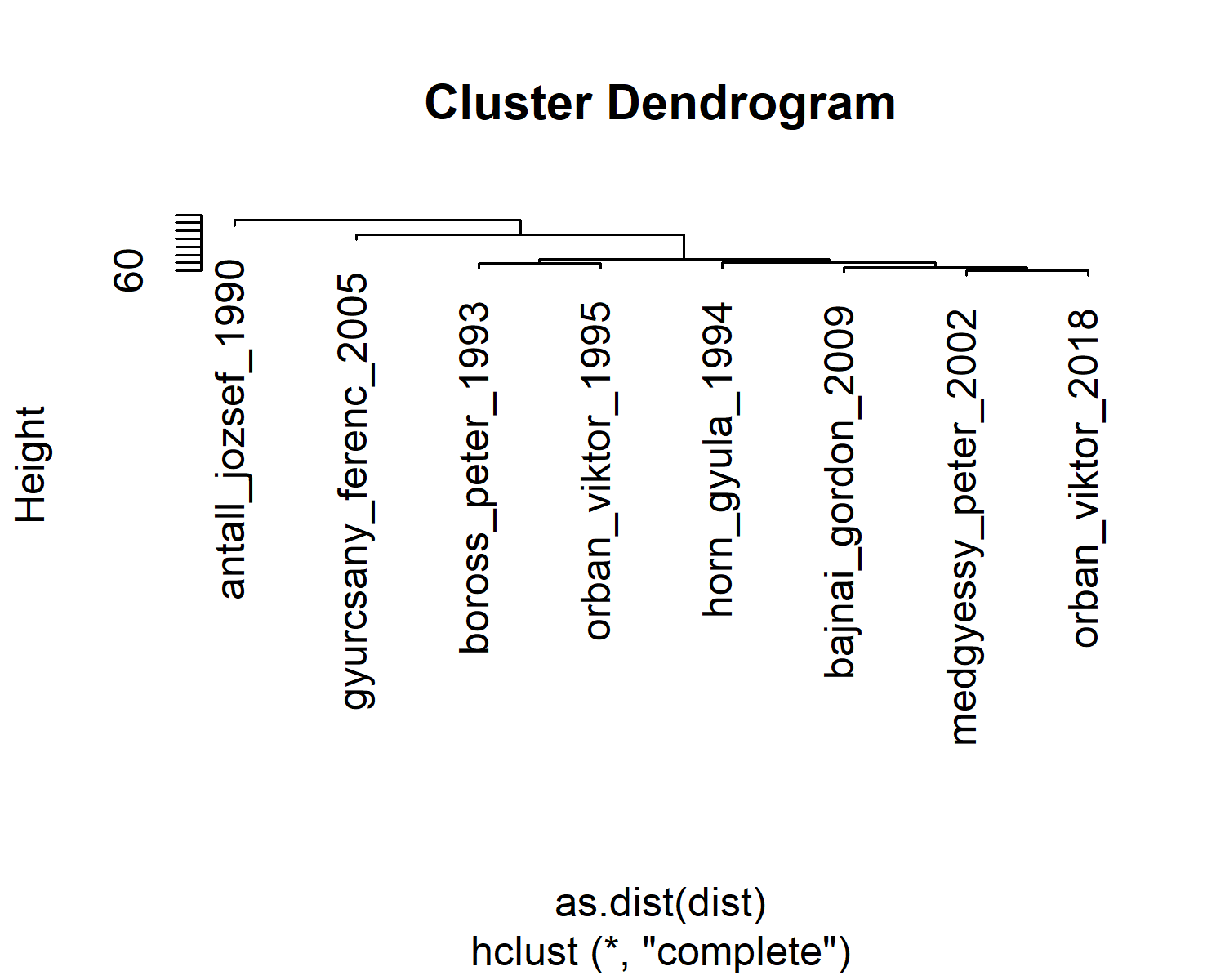


Figure 5.2: A dokumentumok csoportosítása a távolságuk alapján

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 haszáljuk, melynek a bemenete a dfm. A függvény argumentumában a target = után kell szerepletnük, 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)

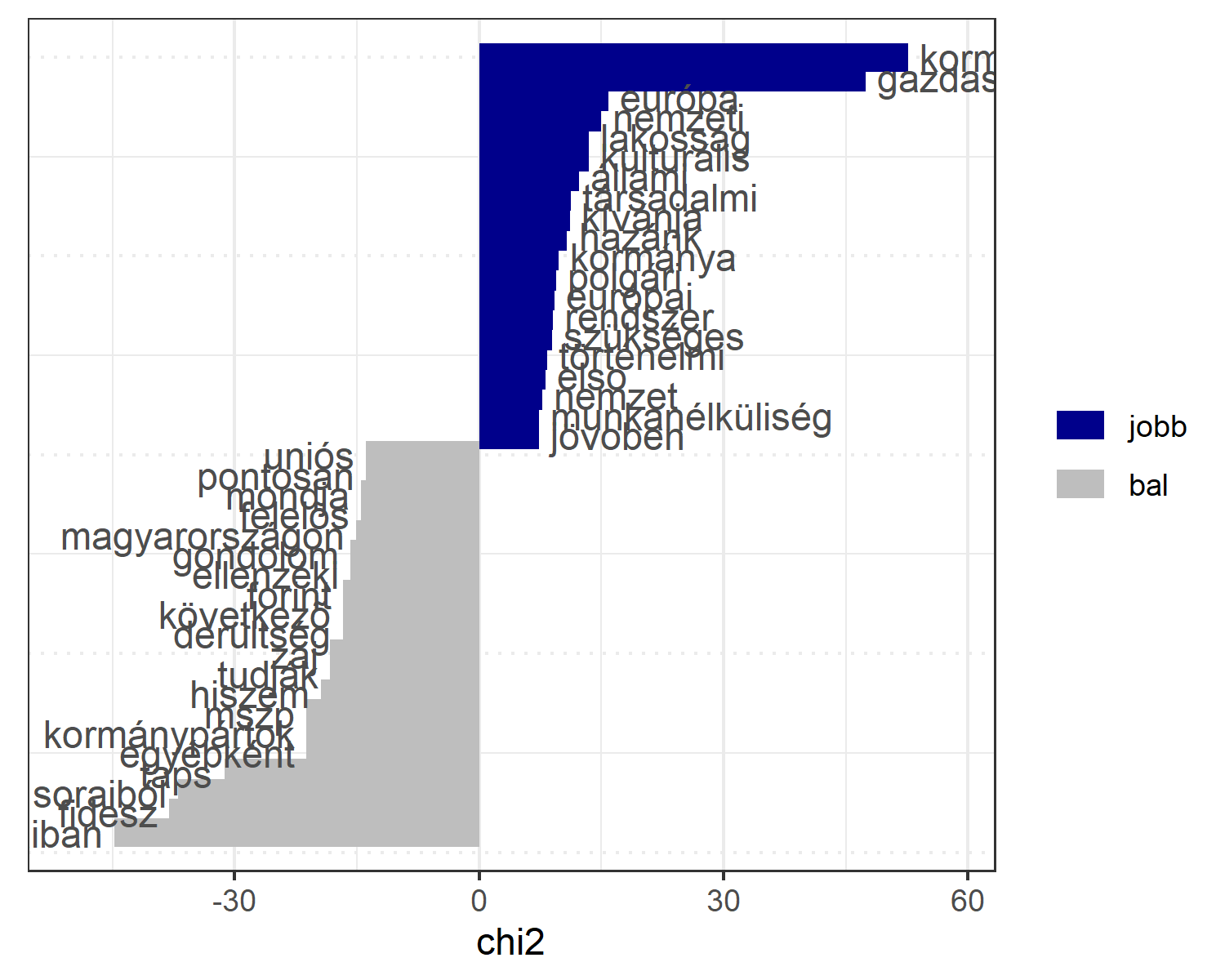


Figure 5.3: A korpuszok legfontosabb kifejezései

Ha az egyes miniszterelnökök beszédeinek leggyakoribb kifejezéseit szeretnénk összehasonlítani, azt a textstat\_frequency() függvény segítségével tehetjük meg, melynek bemnete 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"  
 )

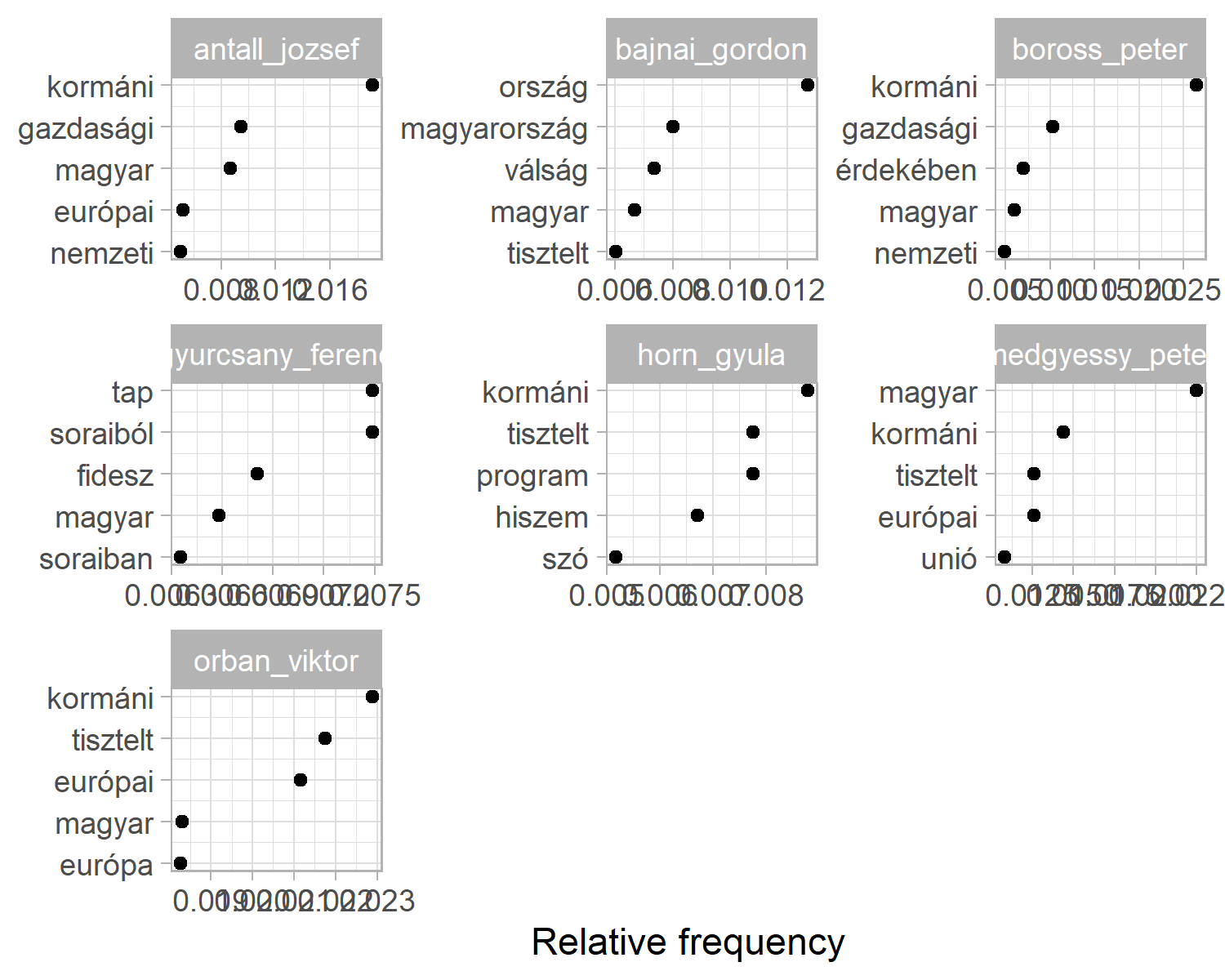


Figure 5.4: Leggyakoribb kifejezések a miniszterelnöki beszédekben

## 5.5 Kifejezések kontextusba helyezése

Arra is lehetőségünk van, hogy egyes kulcszavakat a korpuszon belül szövekörnyezetükben 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 szótár alapú szentiment elemzés egy egyszerű ötleten alapul. Hogyha tudjuk hogy egyes szavak milyen érzelmeket, érzéseket, információt hordoznak, akkor minél gyakoribb egy-egy érzelem kategóriához tartozó szó, akkor a szentiment annél inkább jellemző lesz a dokumentumra amit vizsgálunk. Természetesen itt is jó pár dolognak kell teljesülnie ahhoz hogy az elemzésünk eredménye megbízható legyen. 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 (pl.: egy kulcsszó az “öröm” kategóriába tartozik), csak egyszerűbb módszertannal ([Grimmer and Stewart 2013a](#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, amiről ebben a tankönyvben szó van, 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: nincsen probabilisztikus eleme a számításoknak, mint például a Support Vector alapú osztályozásnál, illetve az emberi szövegkódolásnál előforduló problémákat is elkerüljük így.
* Képesek vagyunk vele mérni a szöveg látens dimenzióit.
* Széles körben alkalmazható, egyszerűen számolható. A politikatudományon és számítogépes nyelvtudományokon belül nagyon sok kész szótár elérhető, amik különböző módszerekkel készültek és különböző területet fednek le (pl.: populizmus, pártprogramok policy tartalma, érzelmek, gazdasági tartalom.)[[10]](#footnote-74)
* Relatíve könnyen adaptálható egyik nyelvi környezetből másikba.

**A módszer lehetéges hátrányai:**

* A szótár hatékonysága és validitása azon múlik hogy mennyire egyezik a szótár és a viszgálni kívánt dokumentum területe. Például jellemző hiba, hogy gazdasági bizonytalanságot szeretnék tőzsdei jelentések alapján vizsgálni a kutatók egy általános szentimet szótár használatával.
* A terület-specifikus szótár építése egy kvalitatív folyamat (lsd. a labjegyzetben), éppen ezért gyakran idő és emberi erőforrás igényes.
* A szózsák alapú elemzéseknél a kontextus elvész (ez gyakran igaz a bigram és trigramok használatánál is) a kulcsszavak esetében. Erre egy triviális példa a tagadás a mondatban: *“nem vagyok boldog”* esetén egy általános szentiment szótár a tagadás miatt félreosztályozná a mondat érzelmi töltését.

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 szótárat, 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.[[11]](#footnote-75)

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

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. A második korpusz a Magyar Nemzeti Bank angol nyelvű sajtóközleményeiből áll, amin bemutatjuk egy széles körben használt gazdasági szótár használatát.

## 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 cimlapokon szereplő hírek . A summary(), ahogy a neve is mutatja, egy gyors áttekinténtést nyújt a betöltött adatbázisról. Látjuk, hogy 2834 sorbol (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 elékészítő 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 autómatikusan meta adatokként tárolódnak az változóink. Jelen esetben az egyetlen dokument változónk az a dátum lesz a szöveg mellett. A korpusz dokumentum változóihoz a docvars() segíségével 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 dokument-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 Társadalomtudmányi Kutatóközpont CSS-RECENS és a POLTEXTLab kutatói készítették.[[12]](#footnote-78) Két dimenziót tarlamaz (pozitív és negatív), 2299 pozitív és 2588 negatív kulcsszóval. Ez nem számít kirívóan nagynak a szótárak között, mivel az adott kategóriák minél teljesebb lefedése a cél. Azt is látjuk, hogy a kulcsszavak egyszavas tokenek (szóval nem érdemes bigramokat és trigramokat készítenünk a tokenizálás során), illetve nem szótövek (így szótöveznünk sem kell).

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

poltext\_szotar  
#> Dictionary object with 2 key entries.  
#> - [positive]:  
#> - abszolút, ad, adaptív, adekvát, adócsökkentés, adókedvezmény, adomány, adományoz, adóreform, adottság, adottságú, áfacsökkentés, agilis, agytröszt, áhított, ajándék, ajándékoz, ajánl, ajánlott, akadálytalan [ ... and 2,279 more ]  
#> - [negative]:  
#> - aberrált, abnormális, abnormalitás, abszurd, abszurditás, ádáz, adócsalás, adócsaló, adós, adósság, áfacsalás, áfacsaló, affér, aggasztó, aggodalom, aggódik, aggódás, agresszió, agresszíven, agresszivitás [ ... and 2,568 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 12.497973  
#> 3 0 0   
#> 4 21.1044299 6.449036  
#> 5 11.0358129 8.131890

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 12.497973  
#> 3 2006-01-02 0.0000000 0.000000  
#> 4 2006-01-02 21.1044299 6.449036  
#> 5 2006-01-02 11.0358129 8.131890

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 elkeszült 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 : 2.373   
#> Mean :2006-06-28 Mean : 4.074   
#> 3rd Qu.:2006-09-26 3rd Qu.: 6.280   
#> Max. :2006-12-29 Max. :35.648   
#> neg   
#> Min. : 0.000   
#> 1st Qu.: 0.000   
#> Median : 2.037   
#> Mean : 3.528   
#> 3rd Qu.: 5.348   
#> Max. :39.096

Mielőtt vizualizálnánk az eredményt érdemes a napi szintre aggregálni a szentimentet és egy nettó értéket kalkulálni.[[13]](#footnote-79)

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 egyértelmű trendet nem lehet megállapítani és még a 2006-os év végi turbulens belpolitikai események sem feltétlenül jelennek meg markánsan. Ennek az oka abban is kereshető, hogy egy napilap címlapját ritkán dominálja teljes egészében a belpolitika és így a negatív és pozitív szentimentek kioltják egymást. Természetesen messzemenő következtetéseket egy ábra alapján nem érdemes levonni, de az elemzésünk azt mutatha hogy a nyári hónapok alatt kevesebb volt az igazán negatív címlap, ellenben az év eleje és vége tartalmazta a minta alsó szélső értékeit.

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

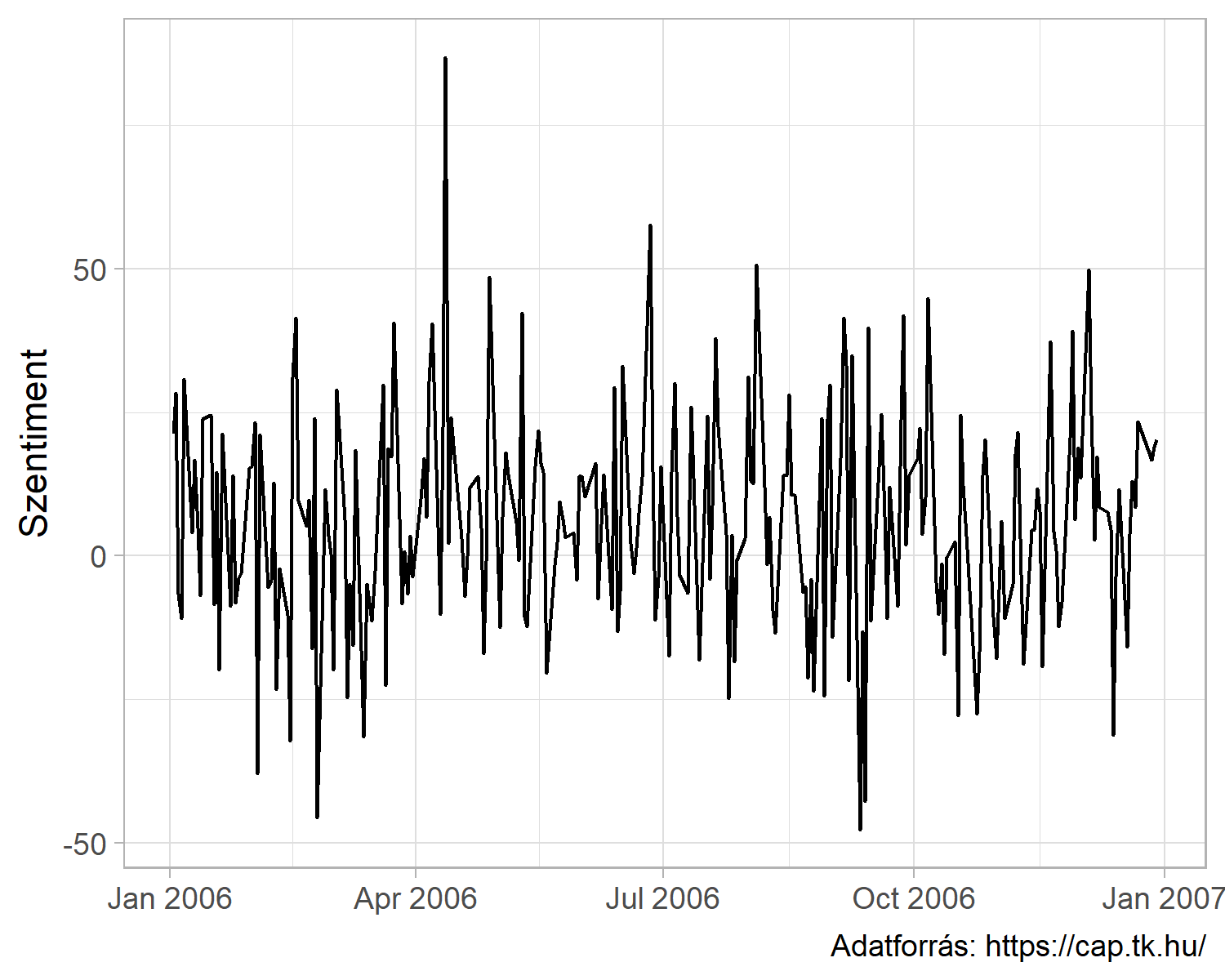


Figure 6.1: Magyar Nemzet címlap szentimentje

## 6.3 MNB sajtóközlemények

A második esettanulmányban a kotextuális 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-loughran2011) ([2011](#ref-loughran2011)) pénzügyi szentimentszótár.[[14]](#footnote-82) A szótár a quanteda.dictionaries csomag részeként elérhető, illetve a tankönyv honlapján is megtalálható.

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 változóbol 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 ellenőrízni, 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 gyüjtjuk 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 megjelnik 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 validálása 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)

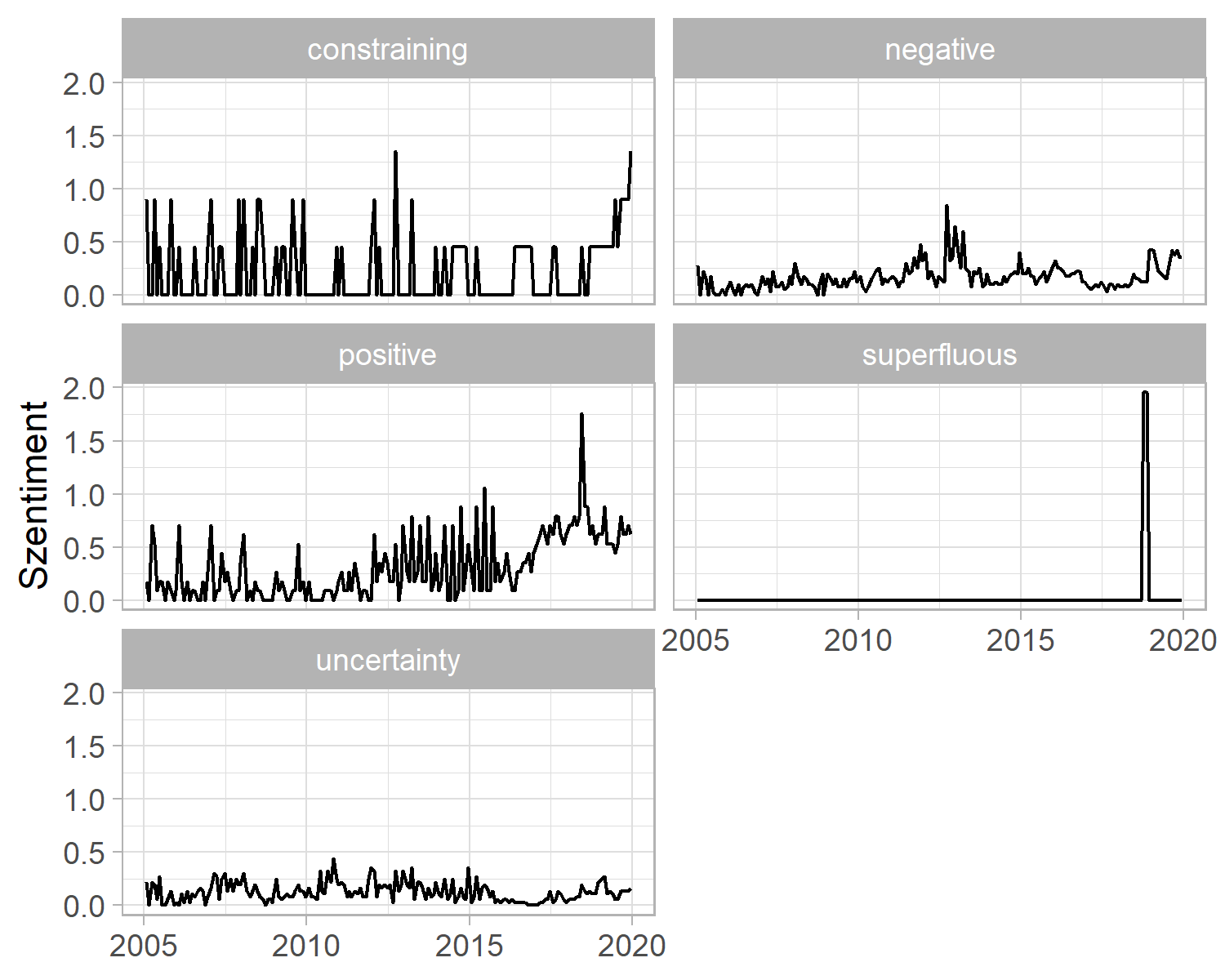


Figure 6.2: Magyar Nemzeti Bank közleményeinek szentimentje

# 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](#ref-tan2011a) ([2011](#ref-tan2011a)). 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](#ref-tan2011a) ([2011](#ref-tan2011a)), [Tikk](#ref-tikk2007) ([2007a](#ref-tikk2007)). 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](#ref-tan2011a) ([2011](#ref-tan2011a)).

## 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](#ref-tikk2007) ([2007a](#ref-tikk2007)), [Burtejin](#ref-burtejin2016) ([2016](#ref-burtejin2016)). 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](#ref-grimmer2013texta) ([2013b](#ref-grimmer2013texta)) . 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](#ref-tikk2007) ([2007a](#ref-tikk2007)). 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 @grimmer2013texta:1.

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 quanteda csomaggal egy korpusz kreálunk, majd abból egy dokumentum-kifejezés mátrixot készítünk a dfm() függvénnyel.

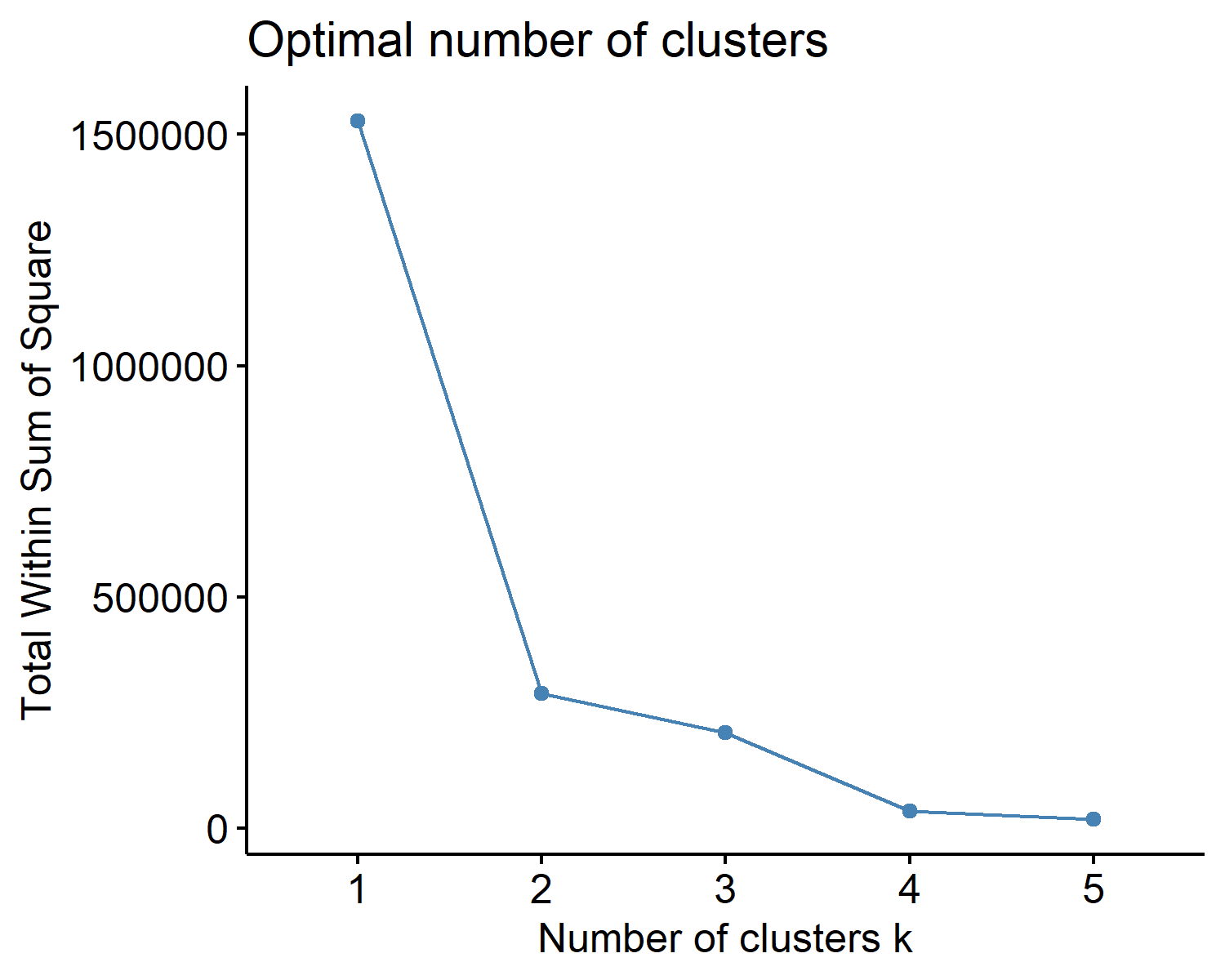
beszedek <- read\_csv("data/miniszterelnokok.csv")  
  
beszedek\_corpus <- corpus(beszedek)  
  
beszedek\_dfm <- dfm(beszedek\_corpus)

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   
#> 2 5

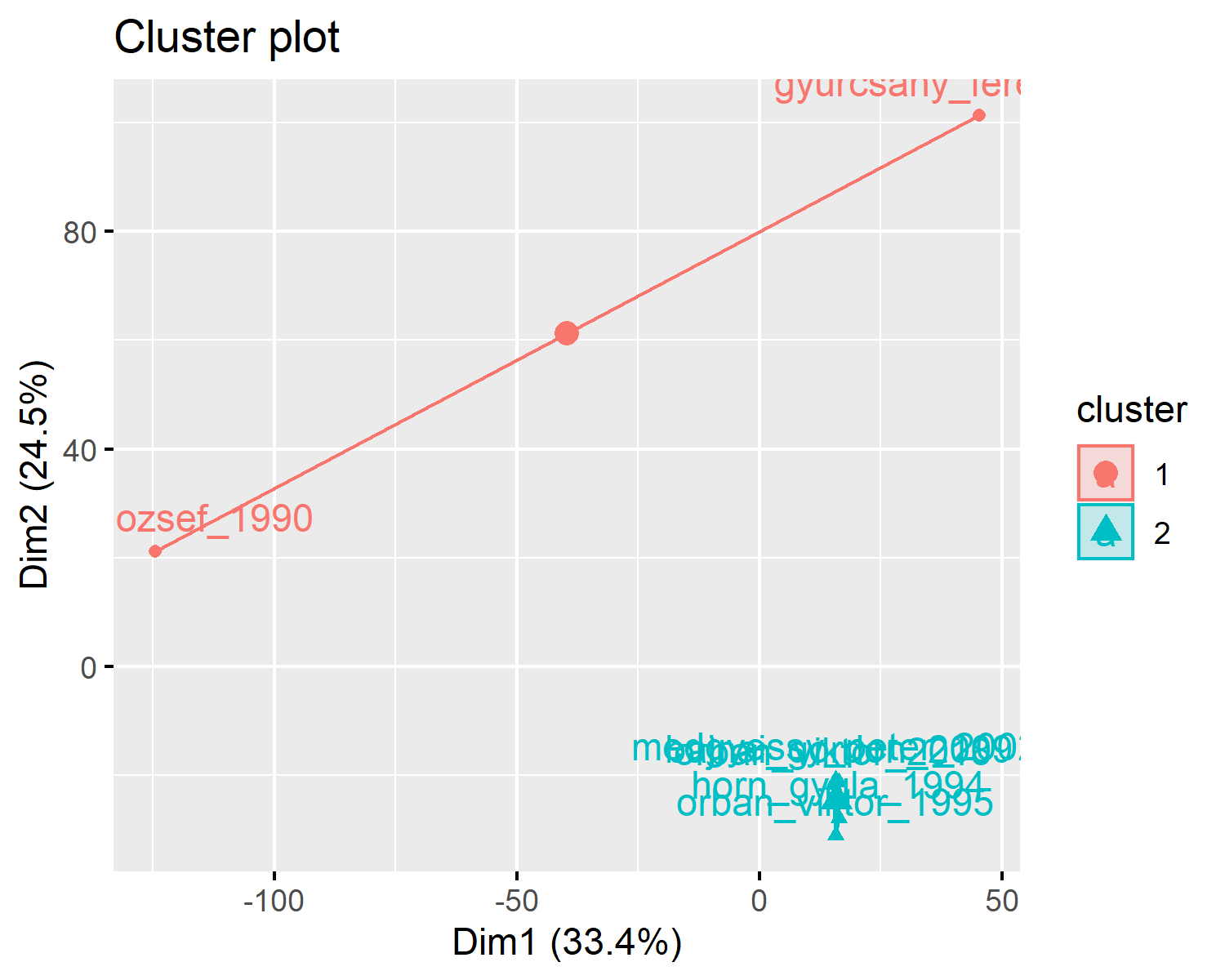
A felügyelet nélküli klasszifikáció nagy kérdése, hogy hány klasztert készítsünk, hogy megközelítsük a valóságot és ne csak mesterségesen kreáljunk csoportokat abban az esetben is amikor ténylegesen nem léteznek. A kvalitatív megközelítések mellett kvantitatív opciók is vannak. A factoextra csomagban több ilyen módszer is van implementálva. A lenti ábra azt mutatja hogy a klasztereken belüli négyzetösszegek hogyan változnak a *k* paraméter változásának függvényében. A lenti ábra alapján az ideális klaszter szám 2.

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



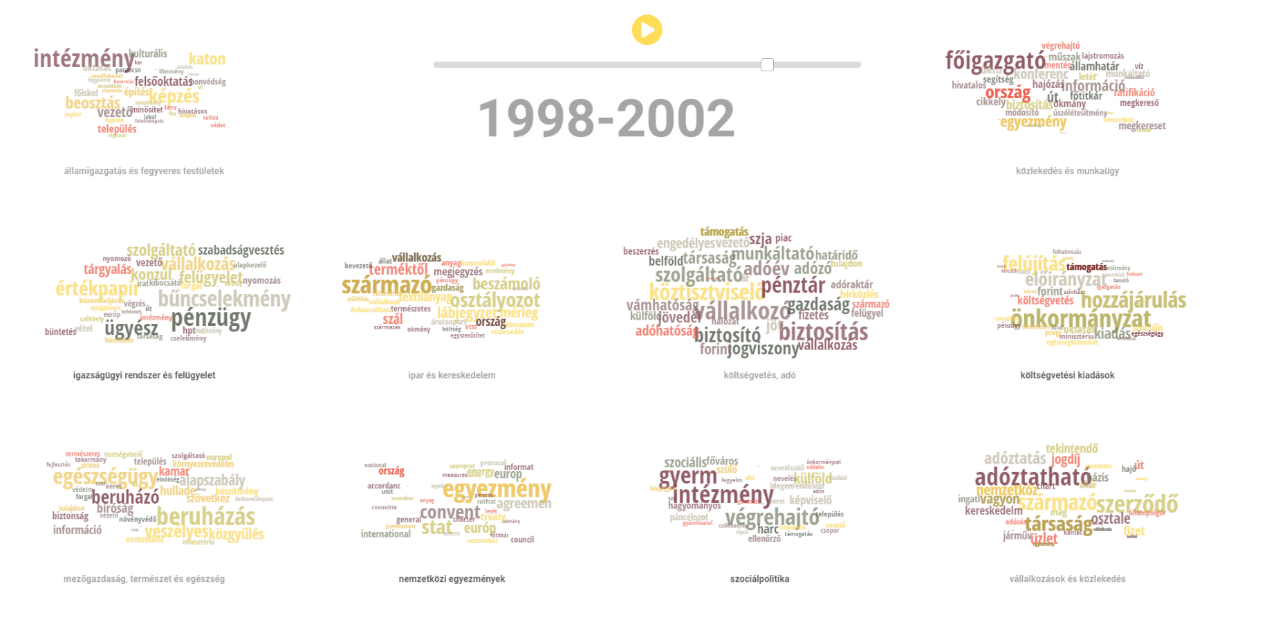
Vizuálisan is megjeleníthetjük a kialakított csoportokat.

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



## 7.2 Látens Dirichlet Allokáció topik modellek[[15]](#footnote-90)

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](#ref-tikk2007a) ([2007b](#ref-tikk2007a)), [Grimmer and Stewart](#ref-grimmer2013texta) ([2013b](#ref-grimmer2013texta)), [Burtejin](#ref-burtejin2016) ([2016](#ref-burtejin2016)) . 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](#ref-blei2003) ([2003](#ref-blei2003)) . 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](#ref-blei2003) ([2003](#ref-blei2003)), [Burtejin](#ref-burtejin2016) ([2016](#ref-burtejin2016)), [Jacobi, Van Atteveldt, and Welbers](#ref-jacobi2016a) ([2016](#ref-jacobi2016a)) .



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. Az következőkben használt fájlok letölthetőek az alábbi github linkről: <https://github.com/poltextlab/text_mining_with_r> A fájlokat töltsük be az R által használt munkakönyvtárba.[[16]](#footnote-94)

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 torvenyek 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 hozzuk létre 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 jogi szövegekről van szó, ezért még egy kis extra szószedetet is készítűnk a felesleges szavakról.

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

Aztán pedig a pipe használatával elkészítjük a token objektumunkat. A szótövesített tokeneket egy külön objektumban tároljuk, mert gyakran előfordul hogy

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 fuggvenyet használjuk, ami egy nagyon hasznos megoldás a csoportosított sorrendbe rendezésre a ggplot 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()

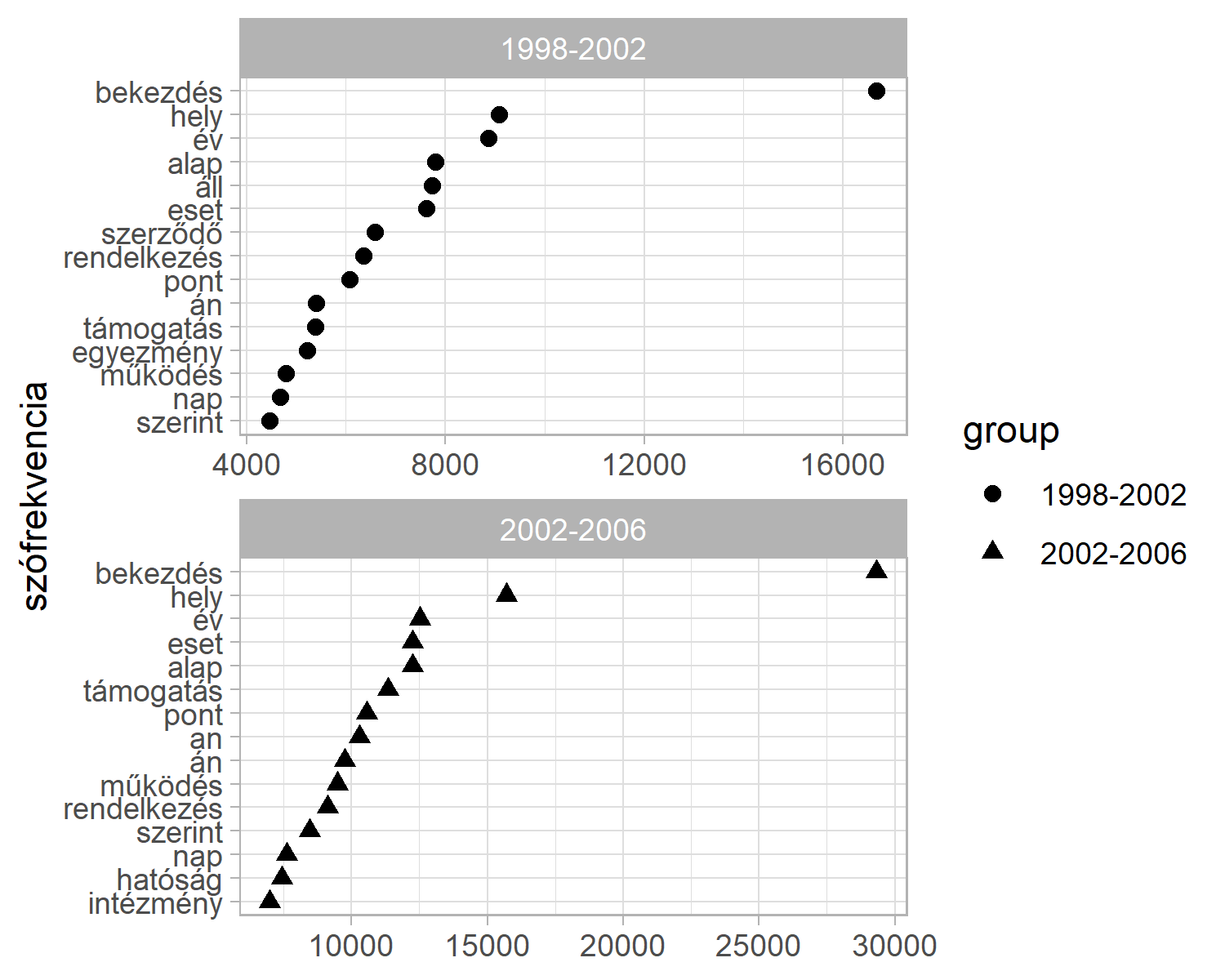


Figure 7.1: A 15 leggyakoribb token a korpuszban

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 ellenőrízzük 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"))

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()

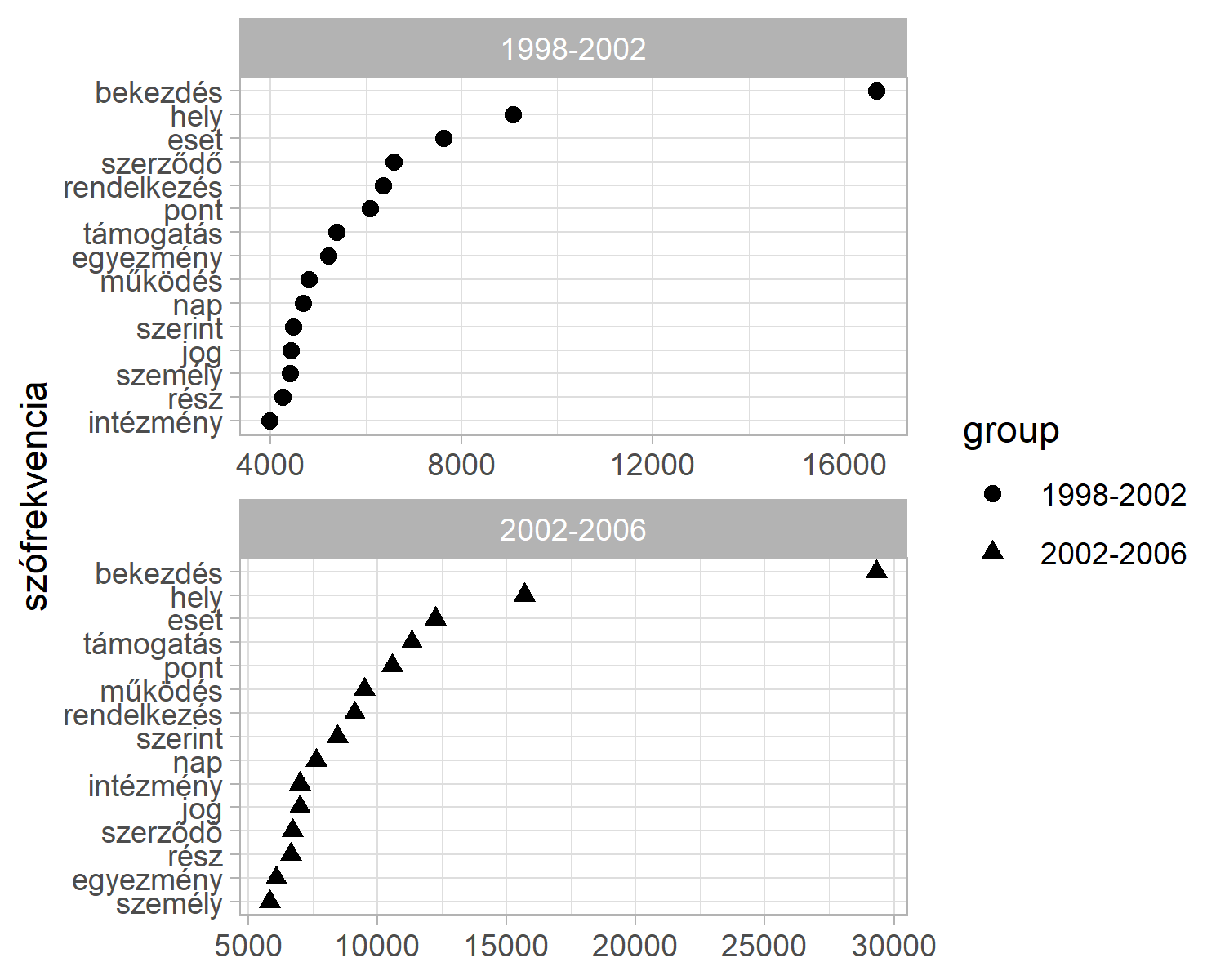


Figure 7.2: A 15 leggyakoribb token a korpuszban, a bovített stop szó listával

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-blei2003); [**Griffiths2002?**](#ref-Griffiths2002); [**Phan2008?**](#ref-Phan2008))

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() a funkció az R véletlen szám generátor magjának beállítására szolgál, ami ahhoz kell, hogy az eredmény, ábra, stb. pontosan reprodukálható legyen. A set.seed() bármilyen tetszőleges egész szám lehet. Kihasználhatjuk hogy minden dokumentumhoz tartozik egy kormányzati ciklus azonosító, mivel ésszerű lehet a feltételezés, hogy különböző parlamentek és kormányok más-más jogalkotási fokusszal rendelkeznek. A dokumentum változók alapján a dfm\_subset()-el tudjuk feldarabolni 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. Bár a k értékének meghatározása kutatói döntésen alapul, és a modell futtatása során bevett gyakorlat a különböző *„k"* értékekkel való kísérletezés, miután elkészült az elemzés a perplexity() funkció segítségével – ahol a *theta* az adott topikhoz való tartozás valószínűsége – lehetőségünk van az elkészült modell kiértékelésére. 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.[^klaszeterzes-2] Az illusztráció kedvéért lefuttatunk 4 LDA modellt az 1998-2002 kormánzyati ciklushoz tartozó dfm-en. Az iterációhoz a purrr csomag map függvényét használjuk (ez a lapply *tidyverse* ekvivalense). Fontos emlékezni arra, hogy minél nagyobb a korpuszunk annál több számítási kapacitásra van szükség (és annál tovább tart a számítás).

[^klaszeterzes-2] <http://brooksandrew.github.io/simpleblog/articles/latent-dirichlet-allocation-under-the-hood/>

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"  
 )

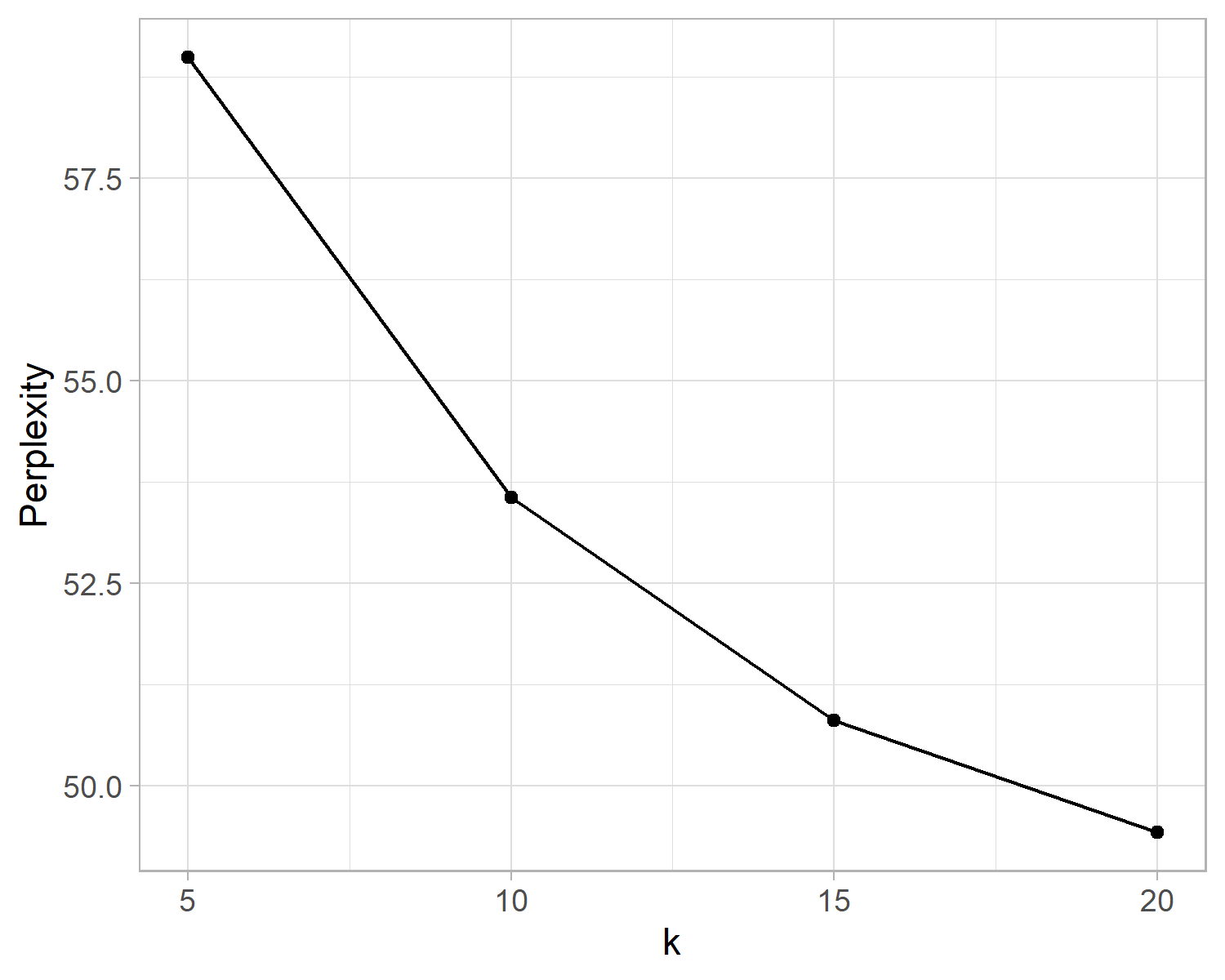


Figure 7.3: Perplexity változása a k függvényében

A perplexity pontszám alapján a 20 topikos modell szerepel a legjobban, de fontos emlékezni arra hogy a megfelelő *k* kiválasztása a kutató kvalitatív döntésén múlik. Ehhez természetesen kvantitatív szempontokat is figyelembe vehetünk, mint például a perplexity indikátor.[[17]](#footnote-100)

A reprodukálhatóság és futási sebesség érdekében a fejezet további részeiben a *k* paraméternek 10-es értéket adunk. Ezzel lefututtatunk 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.[[18]](#footnote-101)

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()

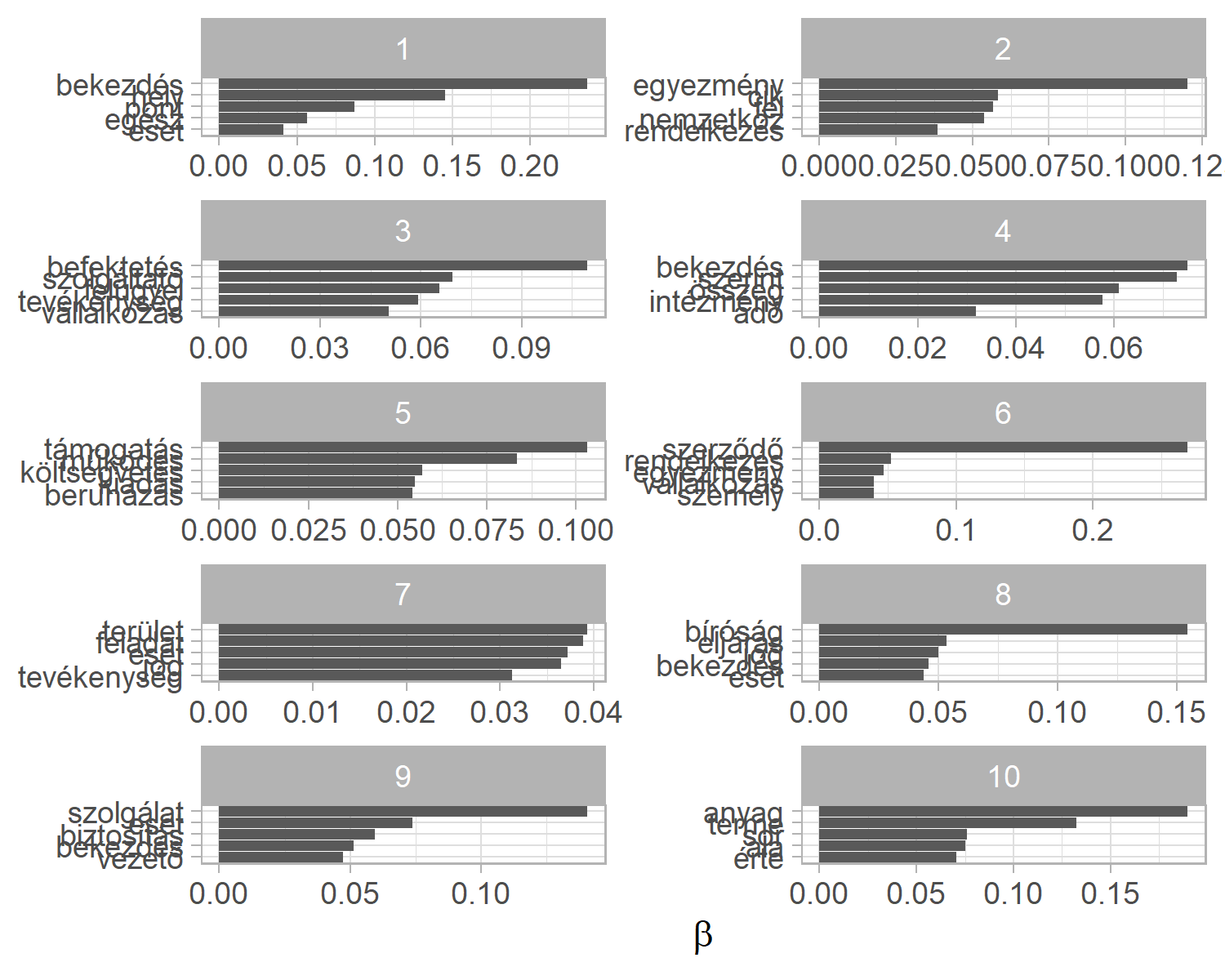


Figure 7.4: 1998-2002 ciklus topikok és kifejezések (VEM mintavételezéssel)

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()

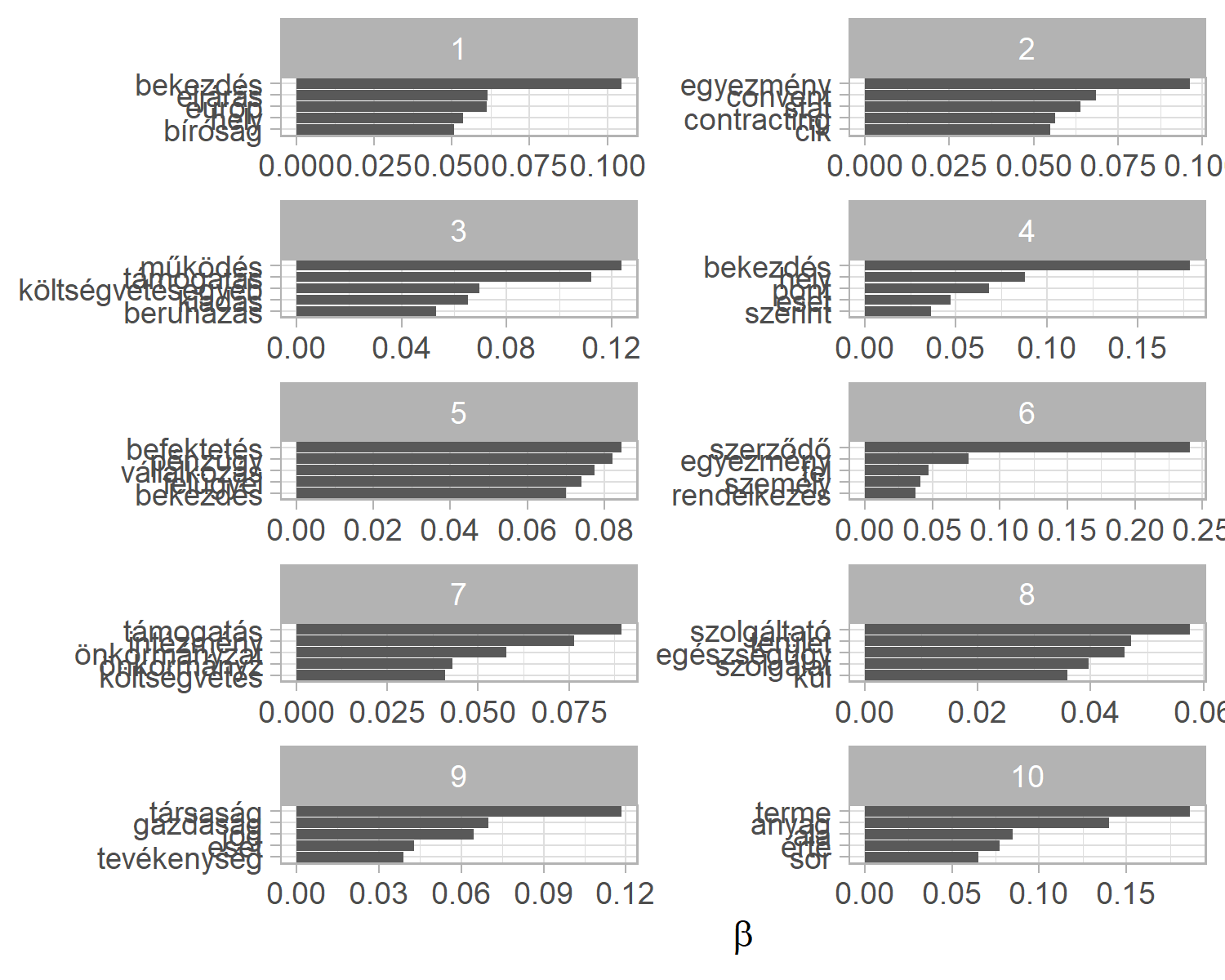


Figure 7.5: 2002-2006 ciklus topikok és kifejezések (VEM mintavételezéssel)

### 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()

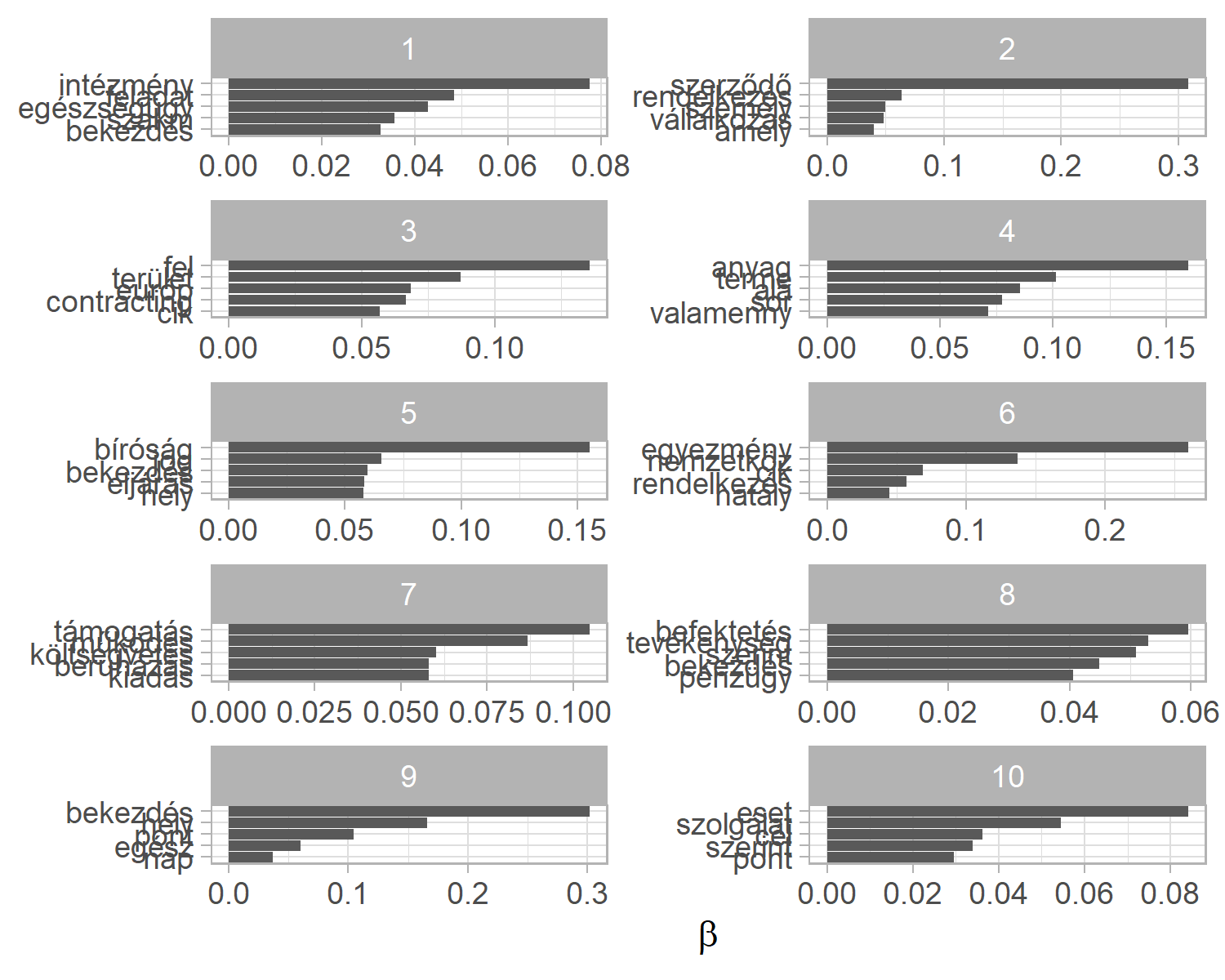


Figure 7.6: 1998-2002 ciklus topikok és kifejezések (Gibbs mintavétellel)

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()

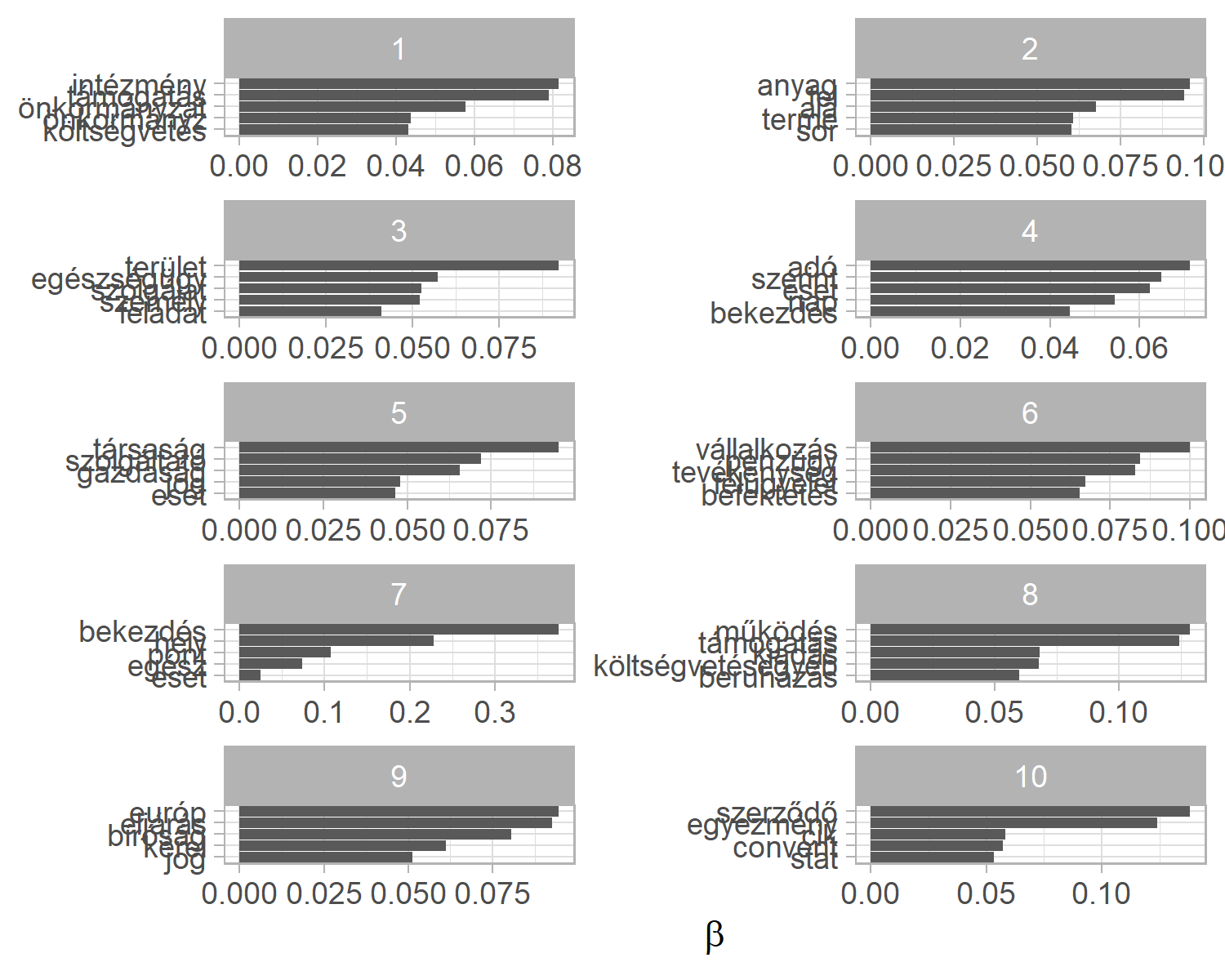


Figure 7.7: 2002-2006 ciklus topikok és kifejezések (Gibbs mintavétellel)

## 7.3 Struktúrális topik modellek

A kvantitatív szövegelemzés elterjedésével együtt megjelentek a módszertani innovációk is és a probabilisztikus topic modellek esetében ez a politikatudomány területéről érkezett. [Roberts et al.](#ref-roberts2014structural) ([2014](#ref-roberts2014structural)) egy kíváló cikkben mutatta be a struktúrális topic modelleket (structural topic models, stm) ahol a fő újítás az az hogy a dokumentumok metaadatai kovariánsként 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 (t*opical 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 dokuemntumokból 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 Comparative Agendas Projekt 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 fügvényekkel tudunk átalakítani és végül vizualizálni a ggplot2 csomaggal. A lenti ábrán az egyes témákhoz tartozó 5 legvalószűbb 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)  
 )

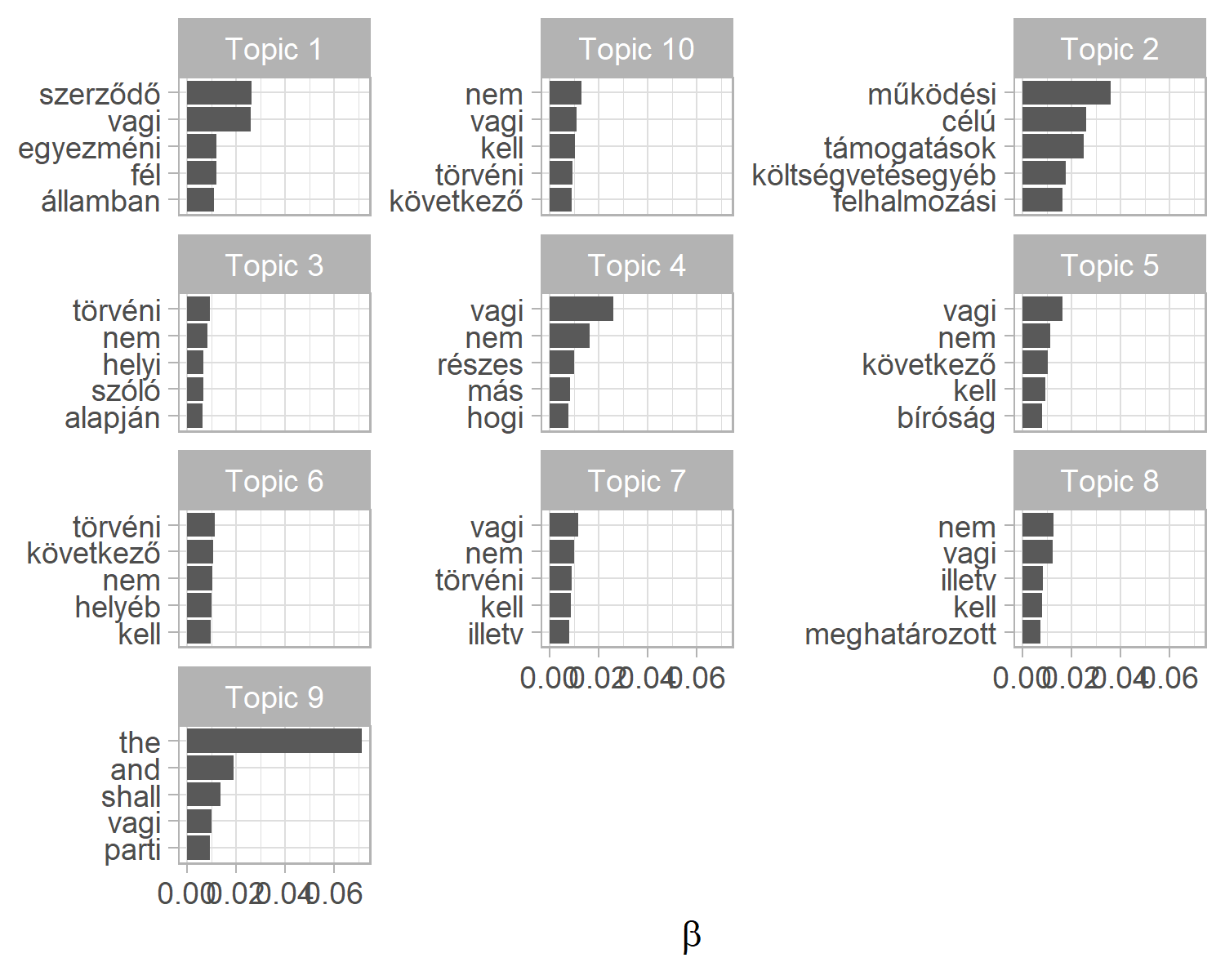


Figure 7.8: Topikonkénti legmagasabb valószínuségu szavak

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.[[19]](#footnote-111)

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")

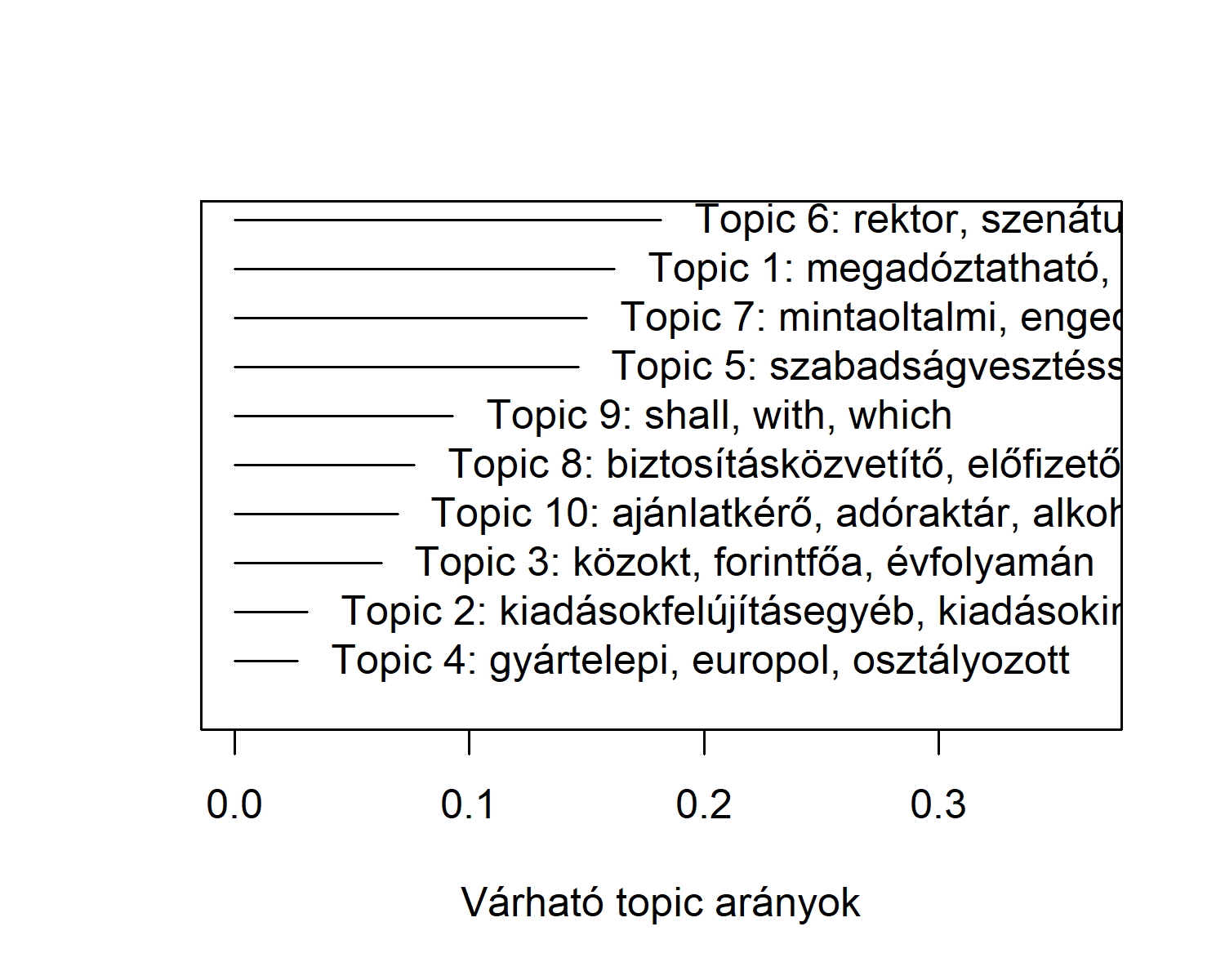


Figure 7.9: Leggyakoribb témák és kifejezések

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))

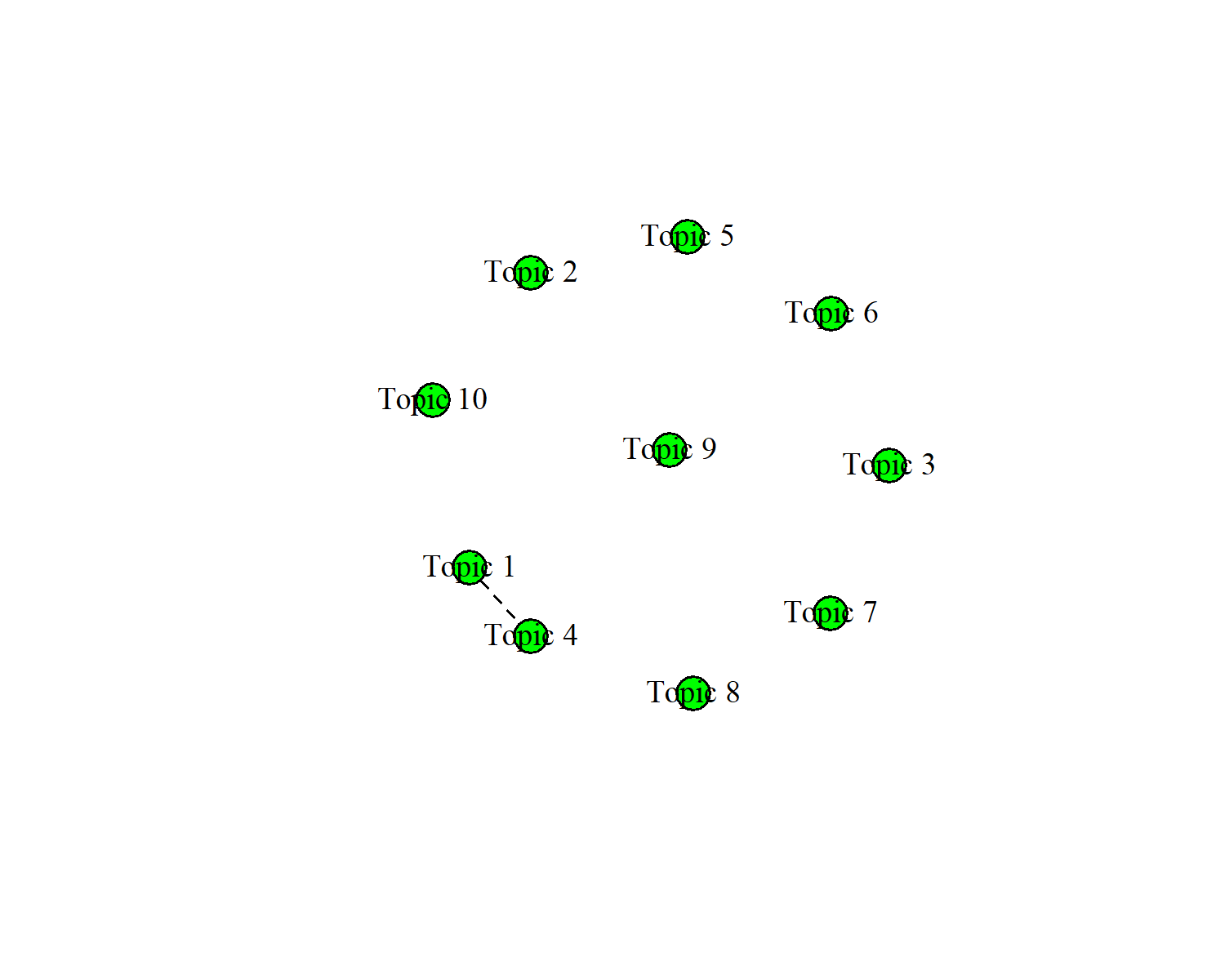


Figure 7.10: Témák közötti korreláció hálózat

# 8 Szóbeágyazások

Az eddigi fejezetekben elsősorban a szózsák (bag of words) alapú módszerek voltak előtérben. Ez a reprezentálása a szövegnek szigorúan véve nem felel meg a valóságnak a kotextuális tartalom elvesztése miatt, de ezt az esetek többségében figyelmen kívül hagyhatjuk. A szóbeágyazáson (word embedding) alapuló modellek viszont kimondottan a kotextuális információt ragadják meg. A szóbeágyazás a topikmodellekhez hasonlóan szintén 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 a modell, 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 klasszkus 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) alapszik, vagyis abból a feltevésből indul ki, hogy a hasonló kifejezések hasonló kontextusban fordulnak elő, valamint mindkettő sekély neurális hálón (2 rejtett réteg) alapuló modell.[[20]](#footnote-117) 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 számításánal, míg a Word2Vec alulsúlyozz 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](#ref-spirlingworda) ([n.d.b](#ref-spirlingworda)).

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

### 8.1.1 GloVe használata magyar média korpuszon

Az elemzéshez a text2vec csomagot fogjuk használni, 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.[[21]](#footnote-118)

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

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 megőrízni, 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.[[22]](#footnote-120) 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)

# saveRDS(mn\_word\_vectors, "data/temp/mn\_word\_vector.RDS")  
  
mn\_word\_vectors <- readRDS("data/temp/mn\_word\_vector.RDS")

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

# 9 Szövegskálázás: felügyelet nélküli és felügyelt megoldások

A szövegskálázás célja a politikai szereplők elhelyezése az ideológiai térben. Ennek felügyelt tipusa 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 𝝷1 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. [([**slapin2008?**](#ref-slapin2008)); ([**hjorth2015?**](#ref-hjorth2015)); ([Grimmer and Stewart 2013a](#ref-grimmer2013text))]

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

A skálázási algoritmusokat egy kicsi korpuszon fogjuk bemutatni. A minta dokumentumok a 2014-2018 parlamenti ciklusban frakcióvezető politikusok egy-egy véletlenszerűen kiválasztott napirend előtti felszólalásai. Ebben a ciklusban összesen 11 frakcióvezetője volt a két kormánypárti és öt ellenzéki frakciónak.[[23]](#footnote-124) A dokumentumokon a rutin előkészítési lépéseket végezzük csak el (tördelések, számok, központozás kitörlése, kisbetűsítés). Természetesen minél alaposabbak vagyunk a szövegek tisztításával, annál pontosabb végeredményt fogunk kapni.

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 fogja használni, amit a szokásos módon a quanteda csomag corpus() függvényével hozunk létre. A leíró statisztikai tááblázatban látszik, hogy a beszédek hosszúsága nem egységes, a leghosszabb 10267 szavas, a legrövidebb pedig 1976. Az átlagos dokumentum hossz az 5135.5714286. A korpusz szemléltető célú, az eddig megszokott módon minél több/hosszabb dokumentummal dolgozunk, annál könnyebb dolga van az algoritmusoknak.

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

Végezetül elkészítjük a dfm mátrixot és a magyar stopszavakat kitöröljük.

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 szófrekfenviát 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

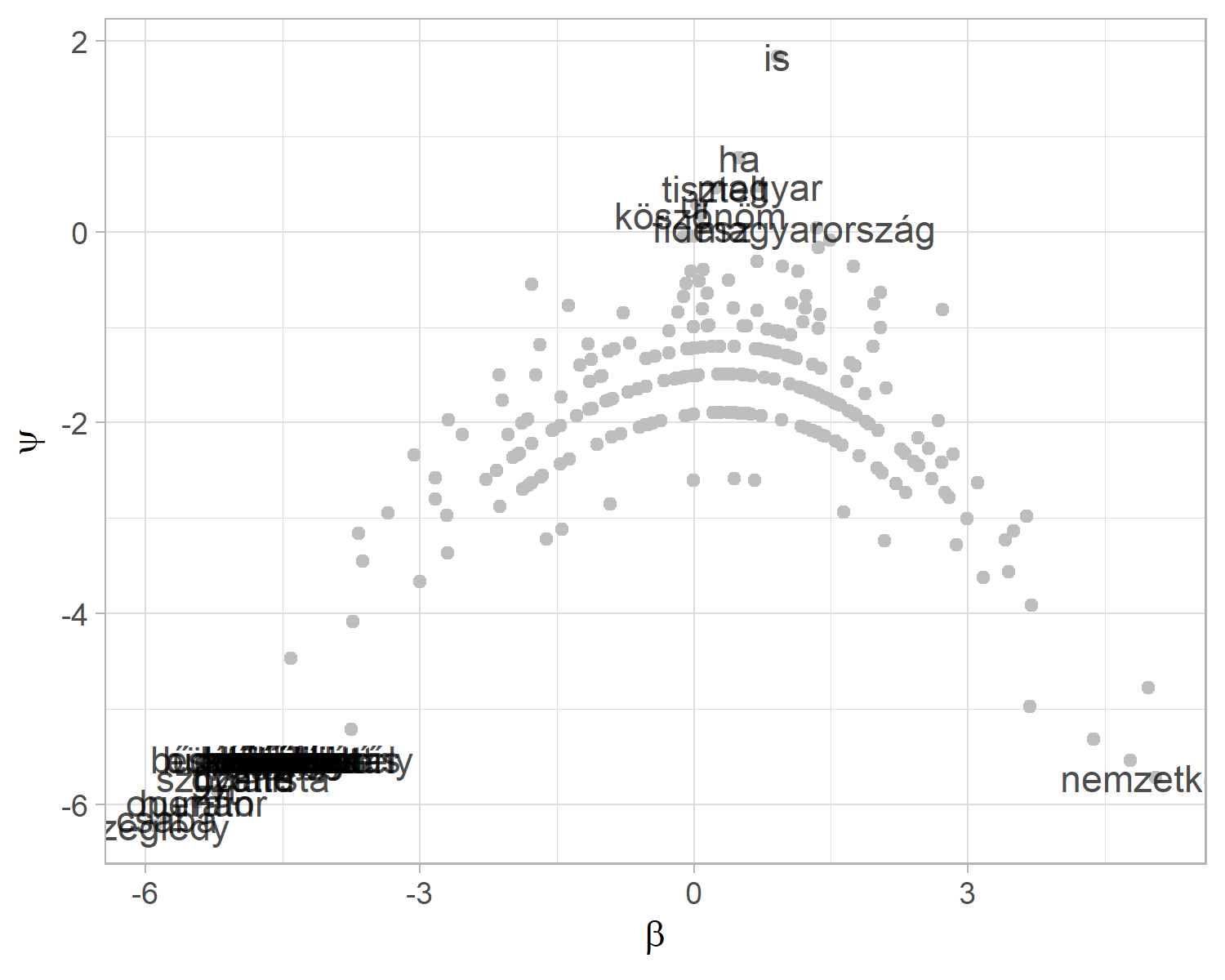
Ez a pozitív értékekre is igaz.

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.[[24]](#footnote-125)

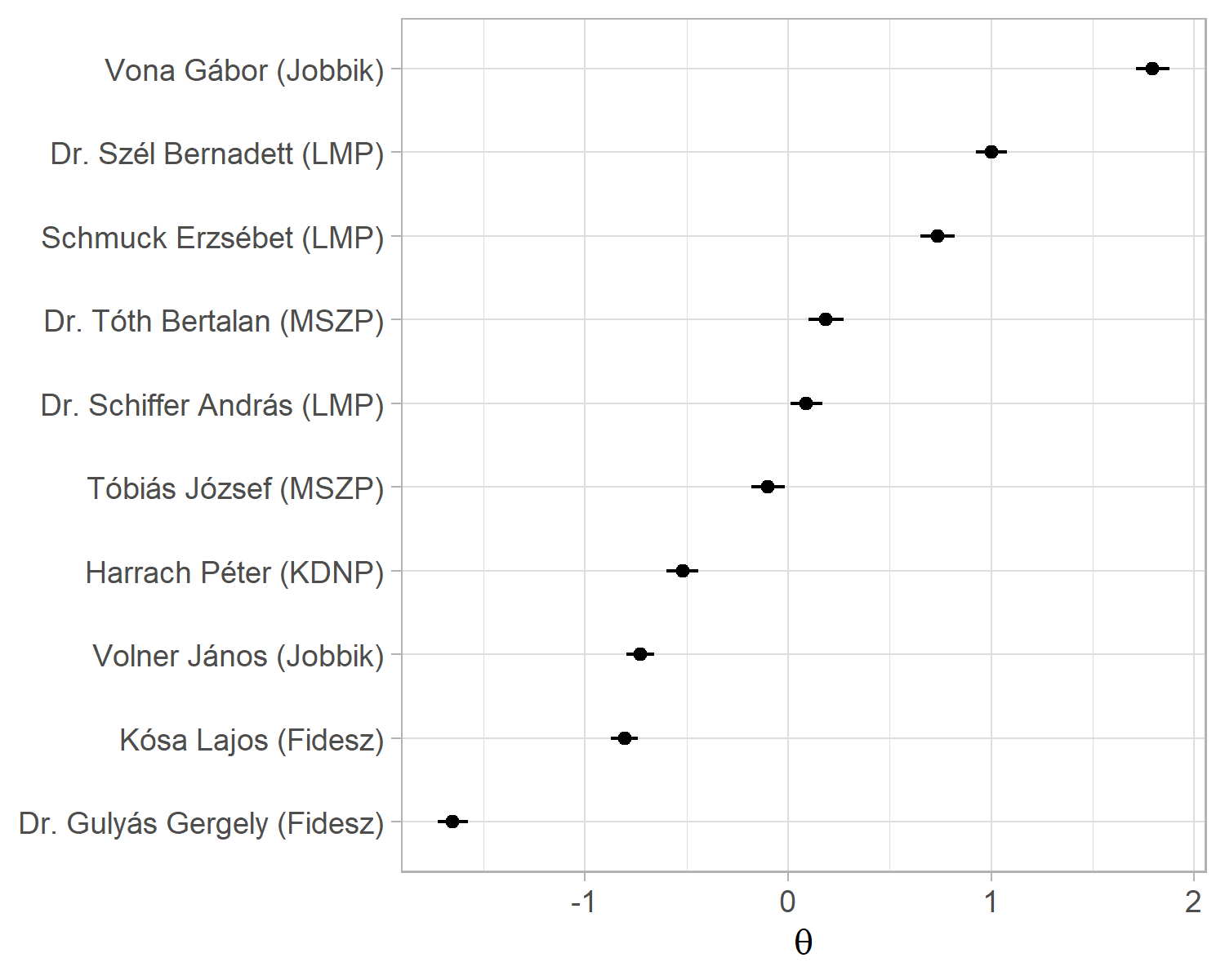
Kihasználhatjuk, hogy a ggplot ábra definiálása közben a felhasznált bemeneti data frame-t különböző szempontok alapján lehet szűrni. így ábrázolni tudjuk a gyakran használt ám semleges szavakat (magas , alacsony ), illetve a ritkább de meghatározóbb szavakat (magas , alacsony ).

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)  
 )



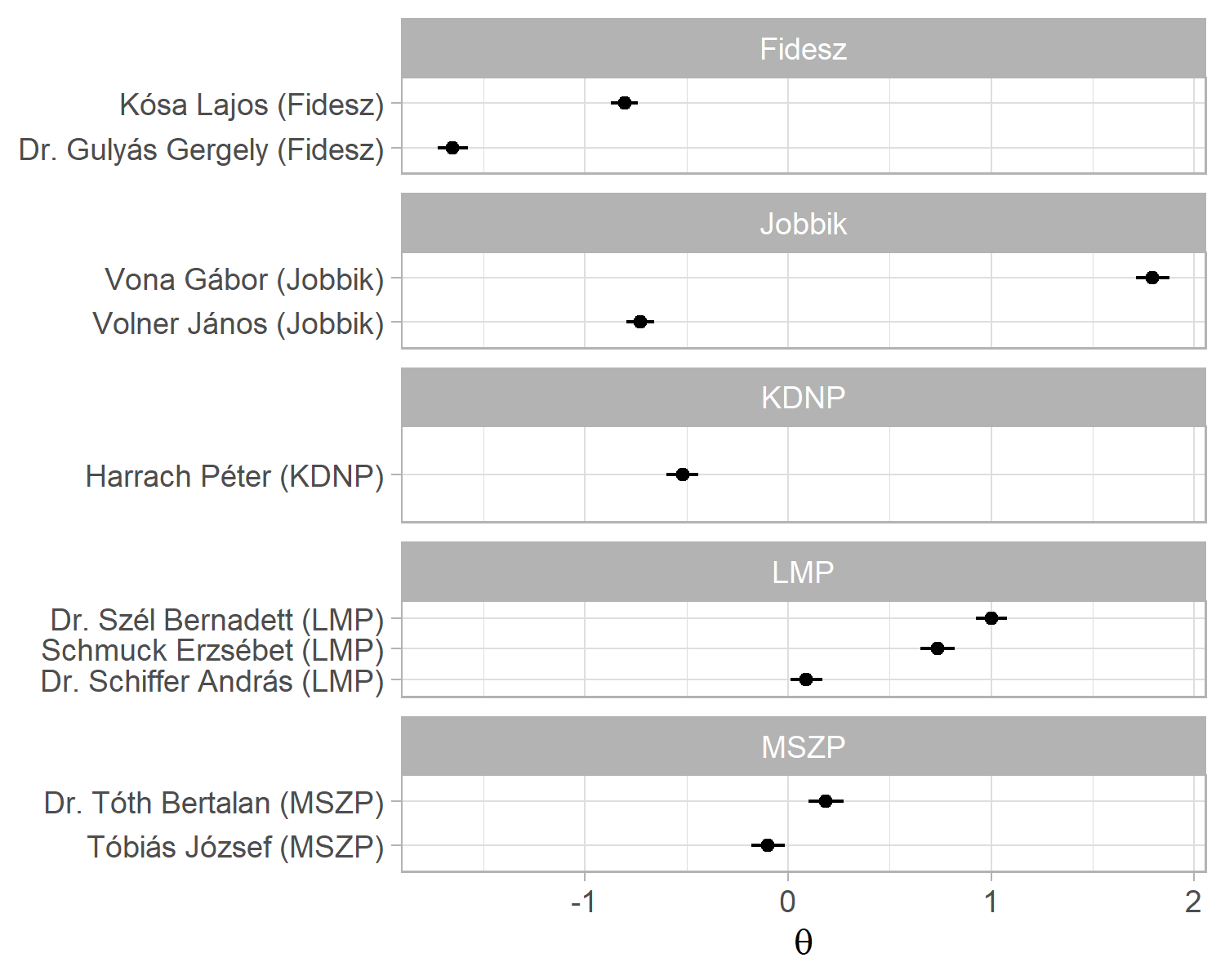
A dokumentumok szintjén is érdemes megvizsgálni az eredményeket. Ehhez a dokumentum szintű paramétereket fogjuk egy data framebe gyűjteni: a ideológiai pozíciót, illetve a beszélő nevét. A vizualizáció kedvéért a párttagságot is hozzáadjuk. A data frame összerakása után az alsó és felső határát is kiszámoljuk 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. Figzeljünk 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*) dimenzó. 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.[[25]](#footnote-130) 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 model 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.

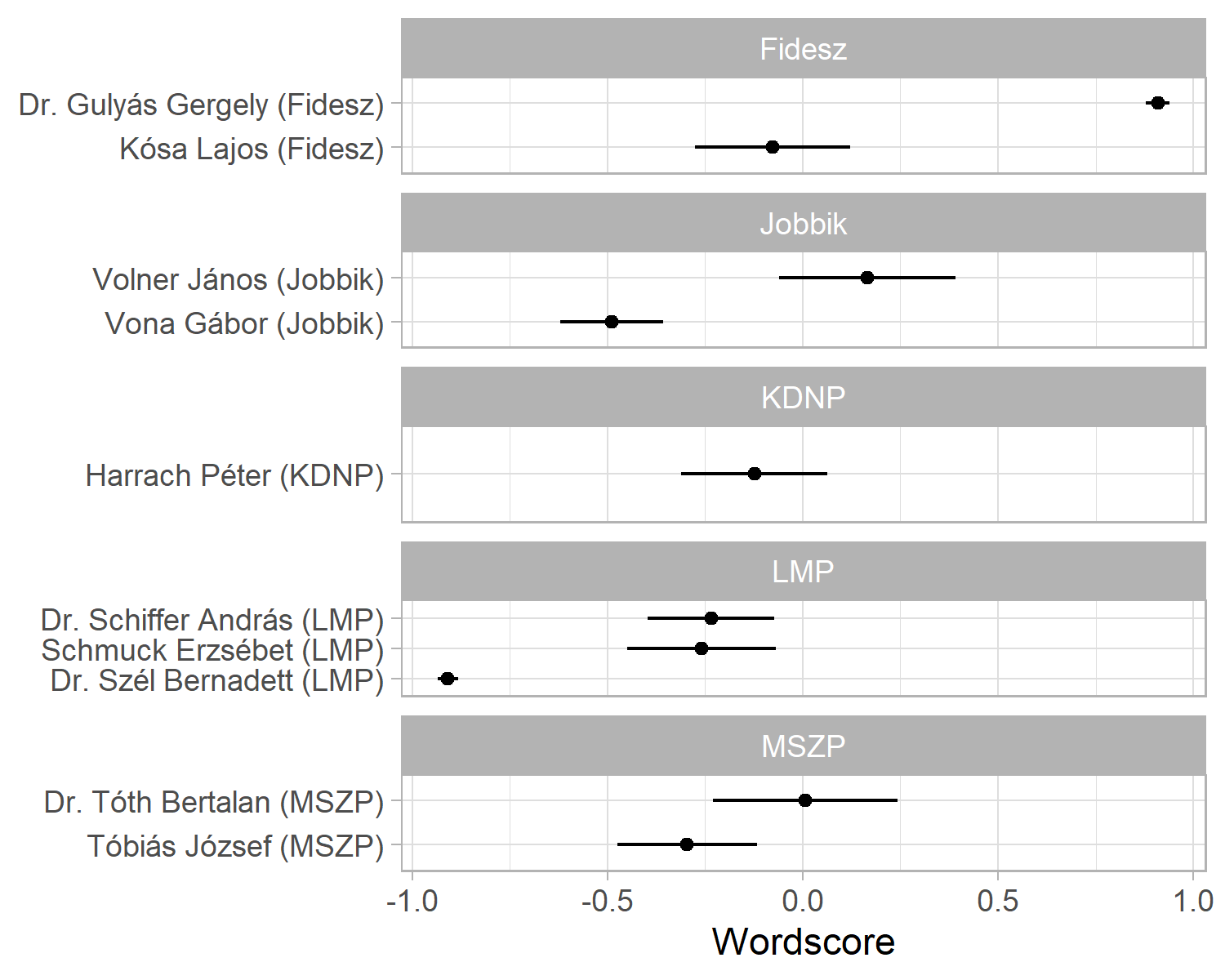
beszedek\_ws\_pred <- predict(  
 beszedek\_ws,   
 newdata = beszedek\_dfm,  
 interval = "confidence")  
  
beszedek\_ws\_pred <- as.data.frame(beszedek\_ws\_pred$fit)  
  
beszedek\_ws\_pred  
#> fit lwr upr  
#> text1 -0.489860579 -0.62138707 -0.35833409  
#> text2 -0.234609623 -0.39658117 -0.07263807  
#> 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

tba

# 11 Természetes-nyelv feldolgozás (NLP)

tizenegyedik fejezet

# 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 kódot 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 reprodukálhatóak 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:

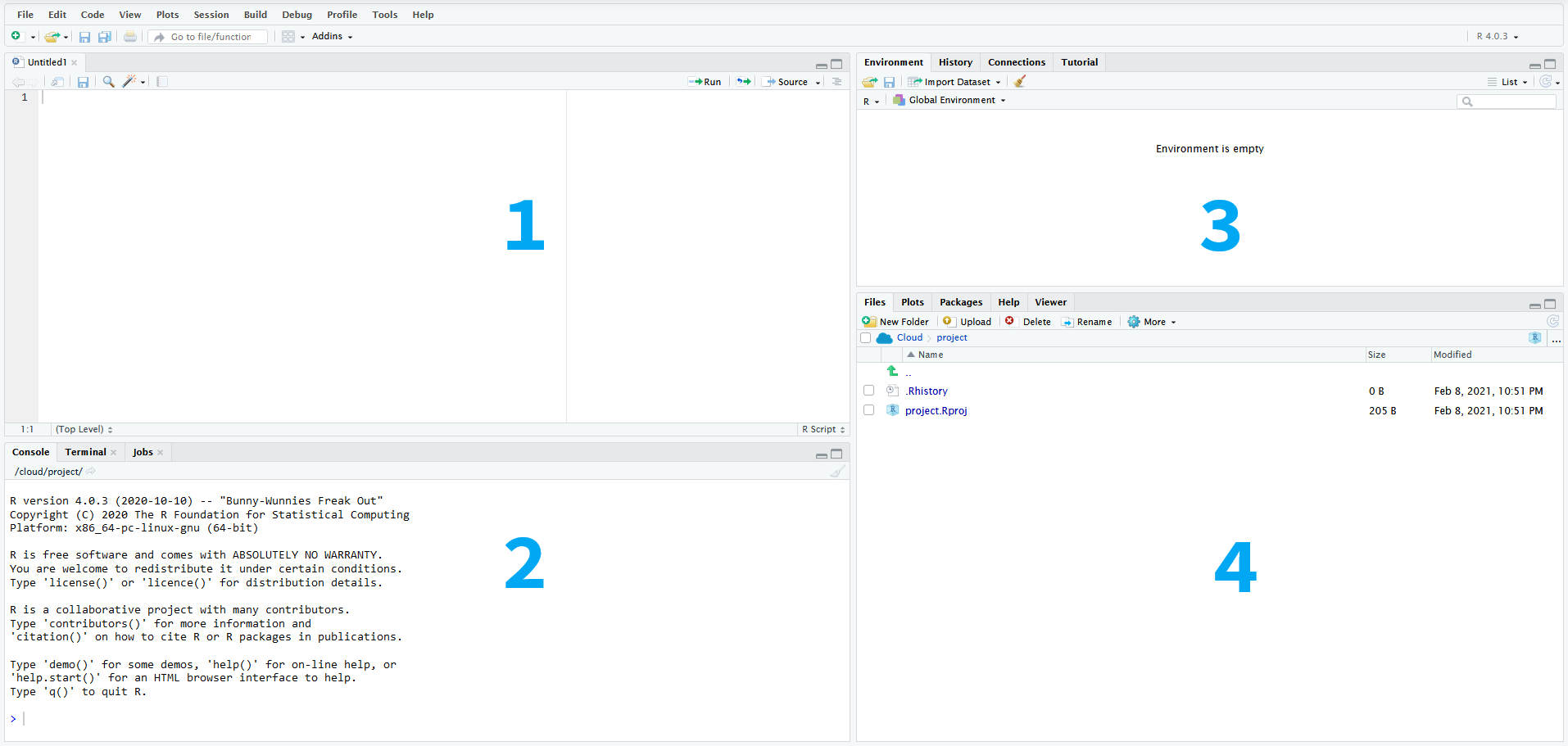


Figure 13.1: RStudio felhasználói felület

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 levő 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 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”* értékre be kell állítani a *„Never”* értéket.

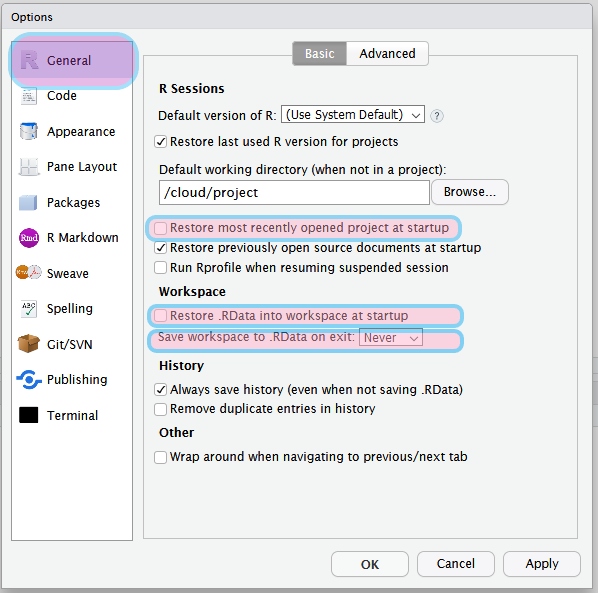


Figure 13.2: RStudio projekt beállítások

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 és 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 scriptben.

A munkafolyamat során létrehozott állományok (ábrák, fájlok) ebbe 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.

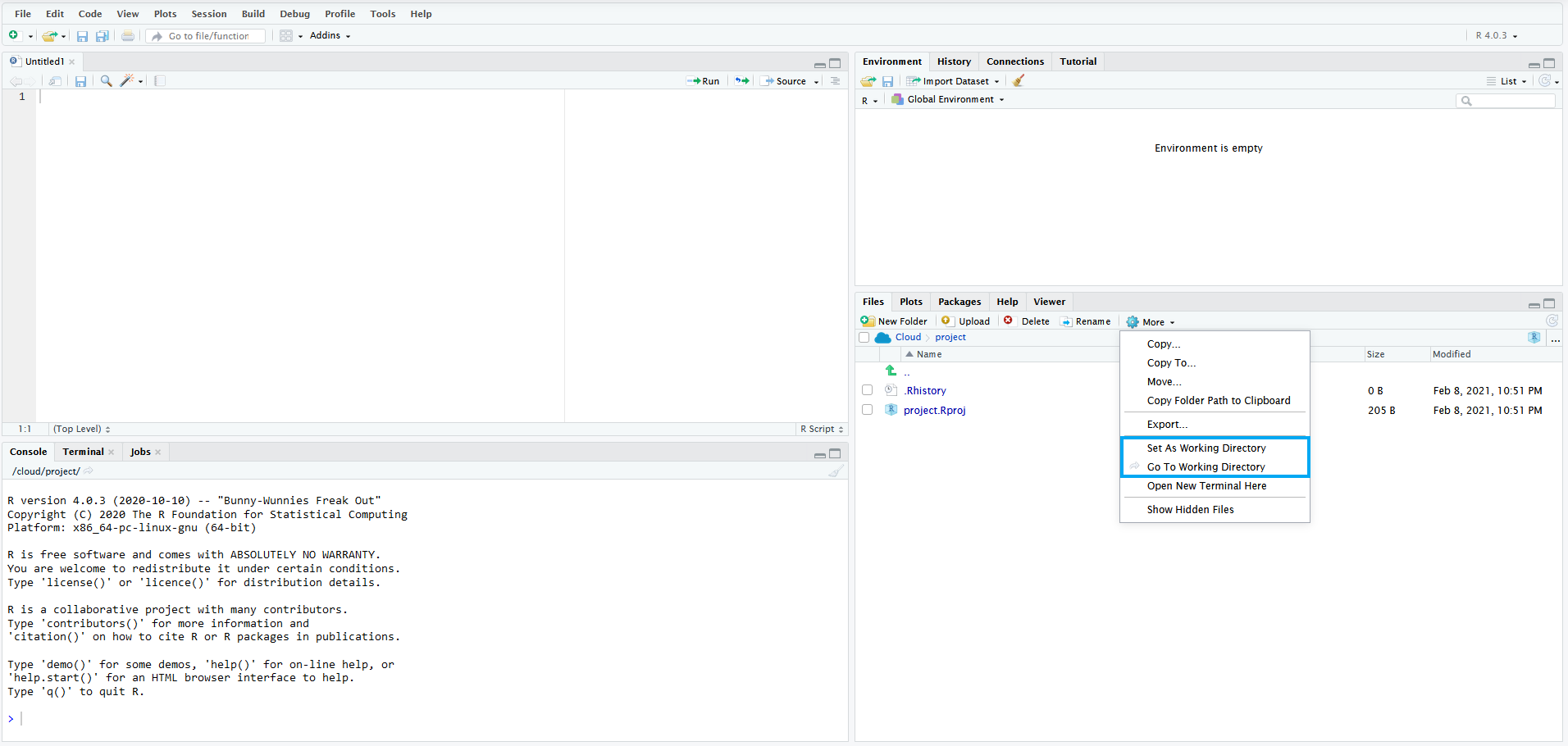


Figure 13.3: Working directory beállítások

A munkafolyamat befejezésére a q() vagy quit() függvényel van lehetőségünk. A munkafolyamat során különböző objektumokat hozunk létre, melyek 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 R-ben objektumokkal dolgozunk, amik a teljesség igénye nélkül lehetnek egyszerű szám vektortok, vagy akár komplex listák, illetve függvények, ábrák.

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)).

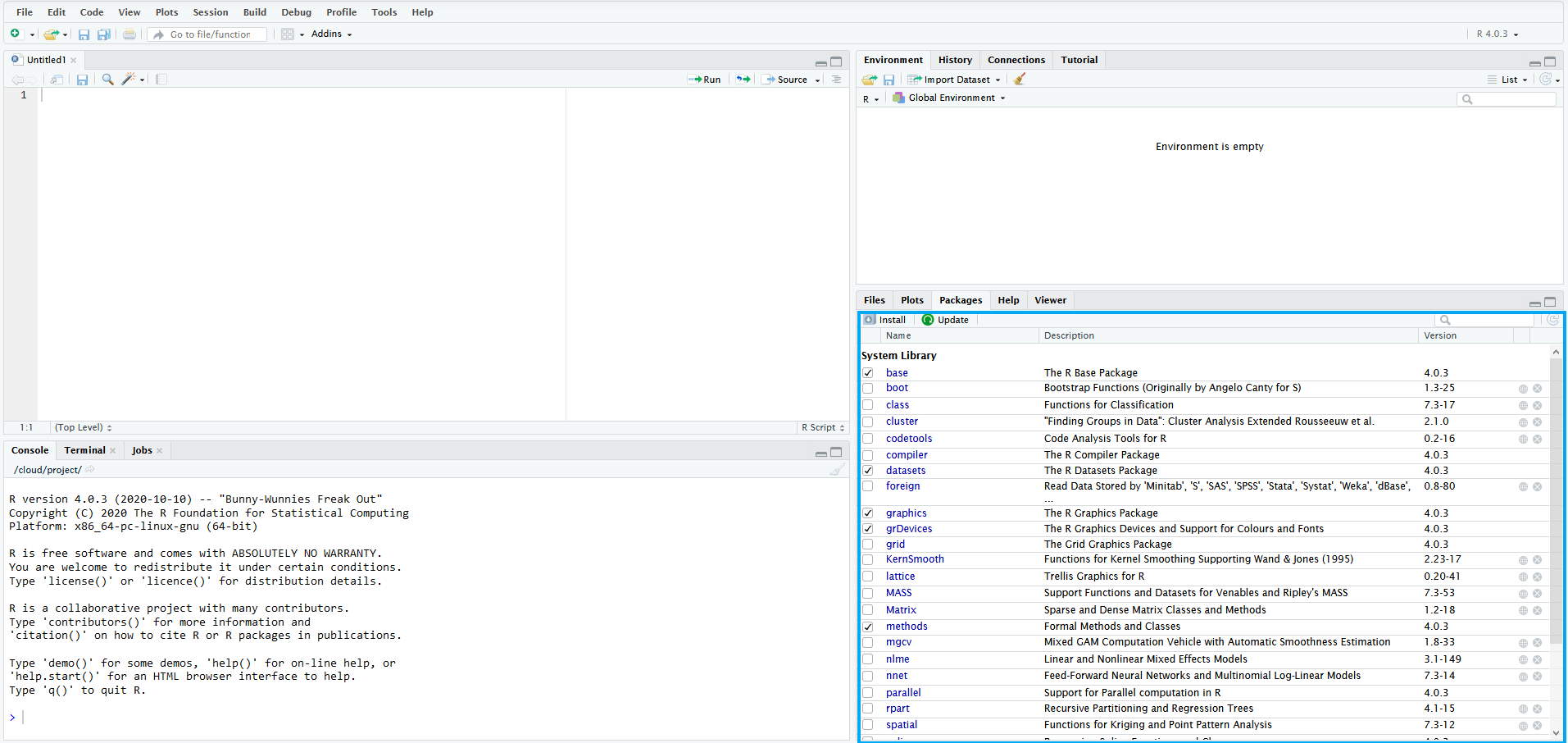


Figure 13.4: Packages fül

### 13.1.5 Objektumok tárolása, értékadás

Az objektumok lehetnek például *vektorok*, *mátrixok* (matrix), *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 összefűzhetjük egymással. Itt egy új objektumot is létrehoztunk, 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 a lefuttatva írathatjuk ki a console ablakba. Habár van print() függvény az R-ben, azt ilyenkor nem szükséges használni.

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 száma), 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 adat táblák (data frame) a statisztikai és adatelemzési folyamatok egyik leggyakrabban használt adattárolási formája. Amikor lehetséges akkor a ‘hosszú’ formátumban használjuk (az R közösség a ‘tidy’ jelzővel illeti), aholtéglalap alakú adatszerkezetek, ahol minden sor egy megfigyelés és minden oszlop egy változó [TIDY CITATION]. 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. A lent 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>

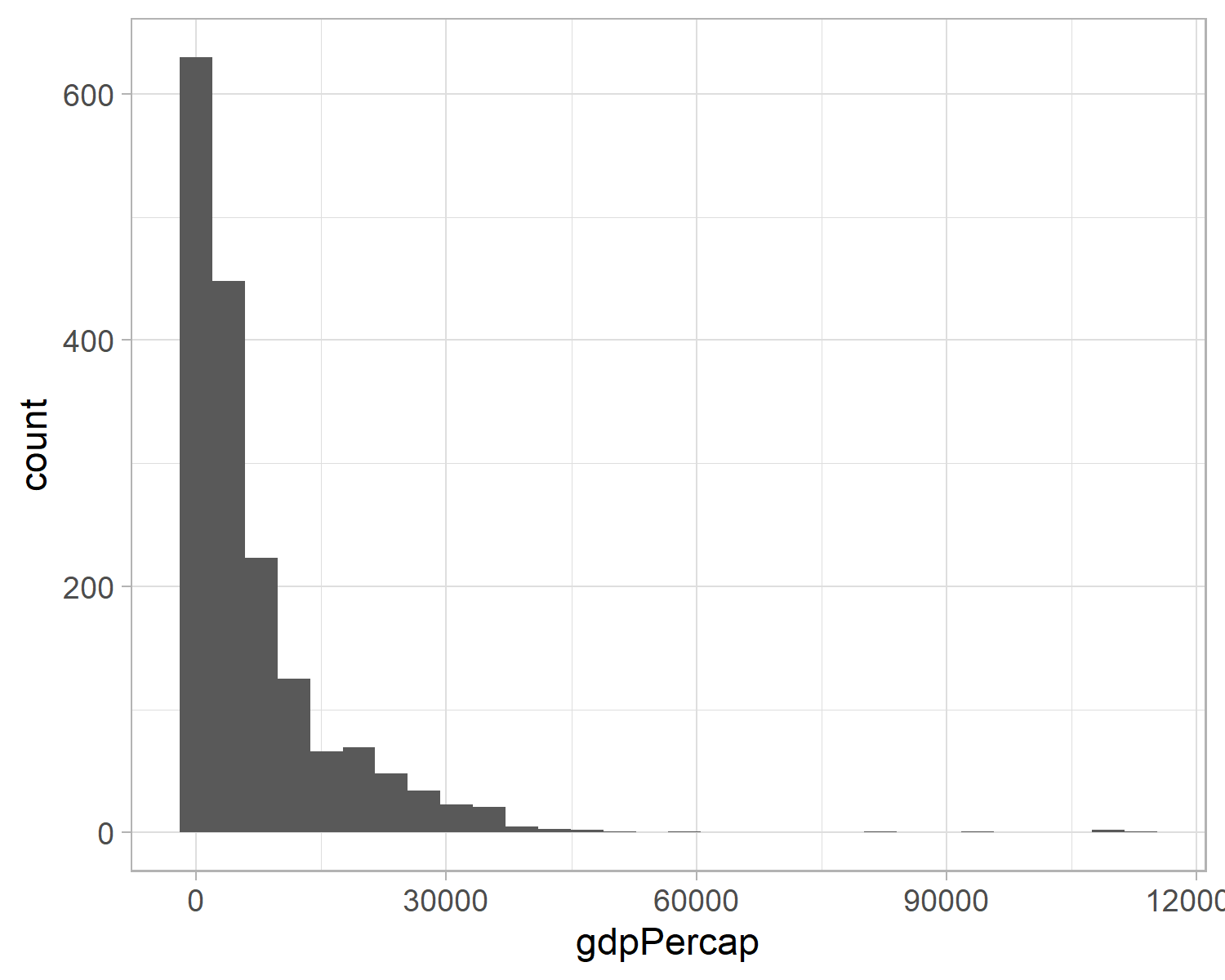
Abban az esetben, ha nem választunk témát, a ggplot2 a következő ábrán is látható alaptémát használja. Ha például a szürke helyett fehér hátteret szeretnénk, alkalmazhatjuk a theme\_minmal()parancsot. Szintén gyakran alkalmazott ábra alap a thema\_bw(), ami az előzőtől az ábra keretezésében különbözik. 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 kiíván változtót vagy változók megadása.

Ezt követi a megfelelő alakzat kiválasztása, attól függően például, hogy eloszlást, változást, adatok közötti kapcsolatot, vagy elétéseket 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>

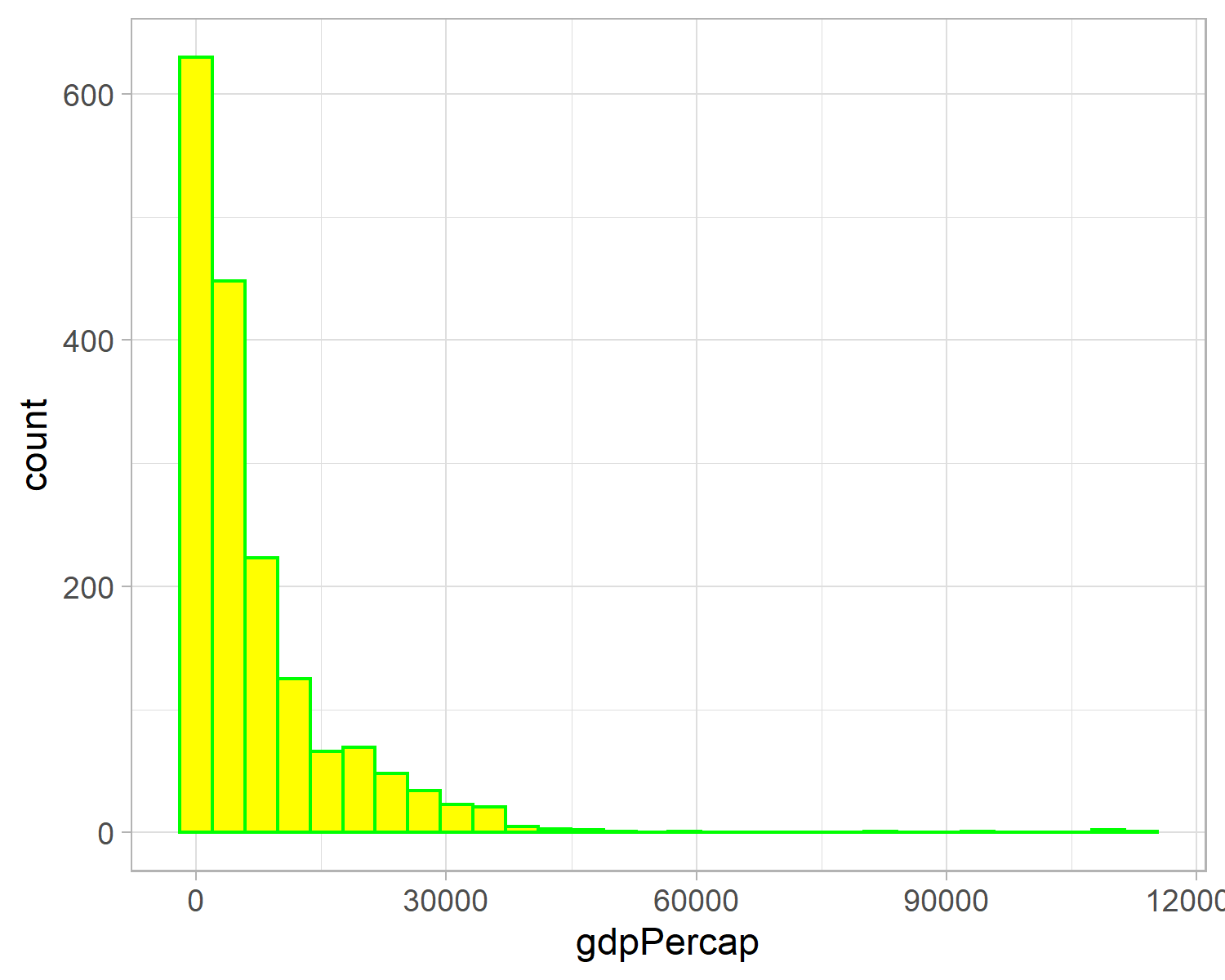
A következőkben a már korábban is használt gapminder adatok segítségével, személetetjük az adatok vizualizálásának alapjait. Először egyszerű alapbeállítások mellett egy histogram típusú vizualizációt készítünk.

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



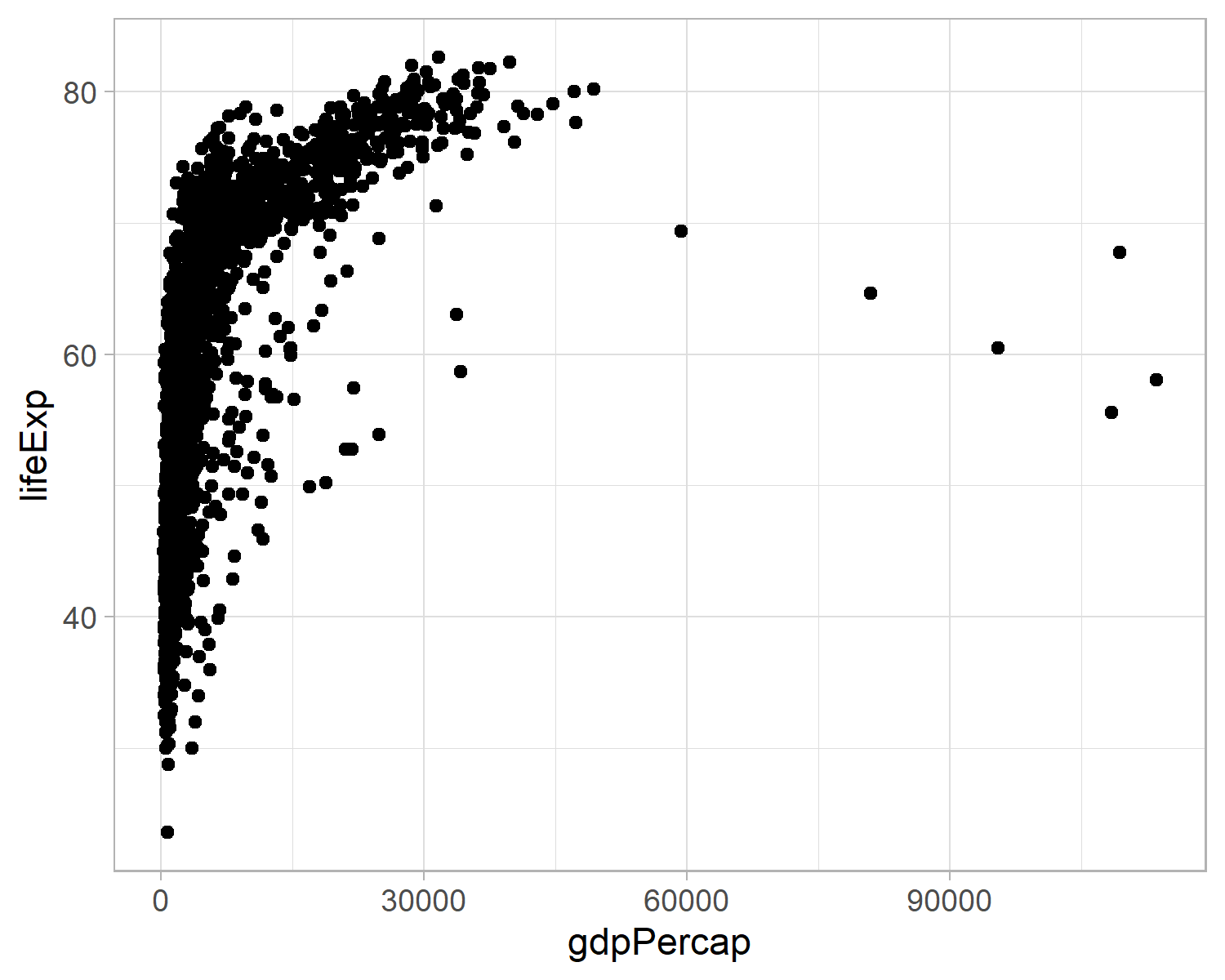
Lehetőségünk van arra, hogy az alakzat színét megváltoztatássuk. A használható színek és színkódok megtalálhatóak a ggplot2 leírásában: <https://ggplot2-book.org/scale-colour.html>

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



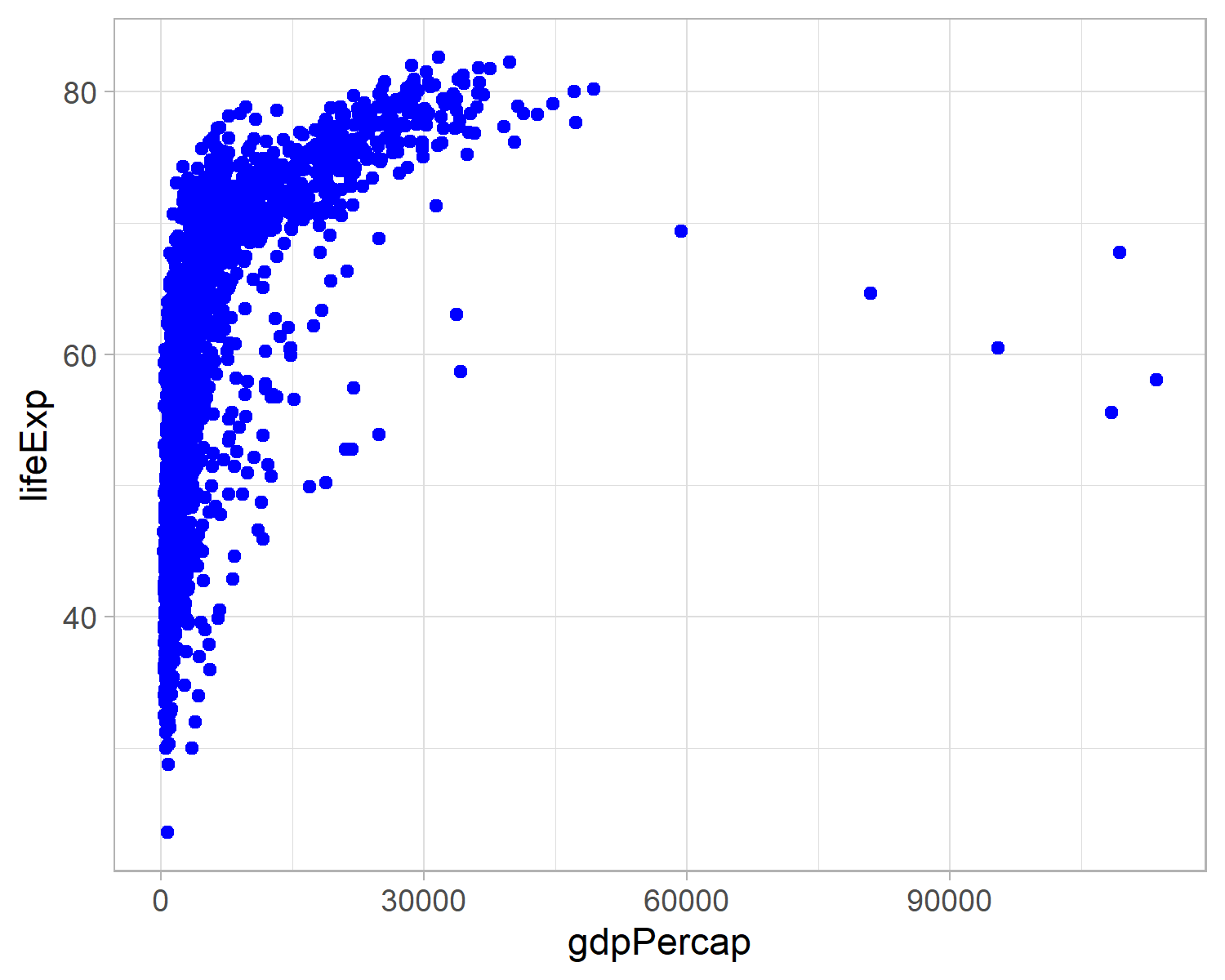
Meghatározhatjuk külön-külön a histogram x és y tengelyén ábrázolni kívánt adatokat és választhatjuk azok pontszerű ábrázolását is.

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



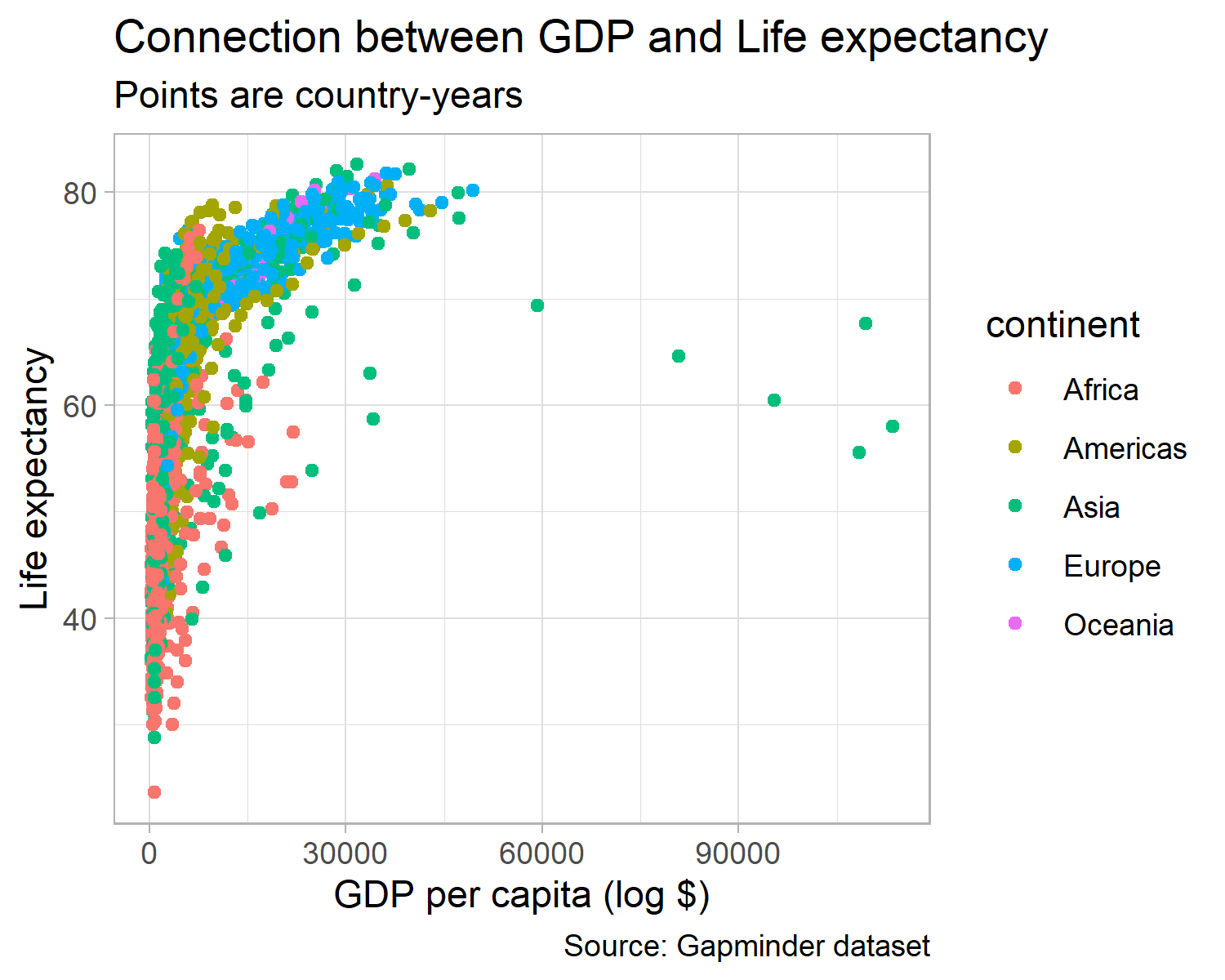
Ahogy az előzőekben, itt is megváltoztathatjuk az ábra színét.

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



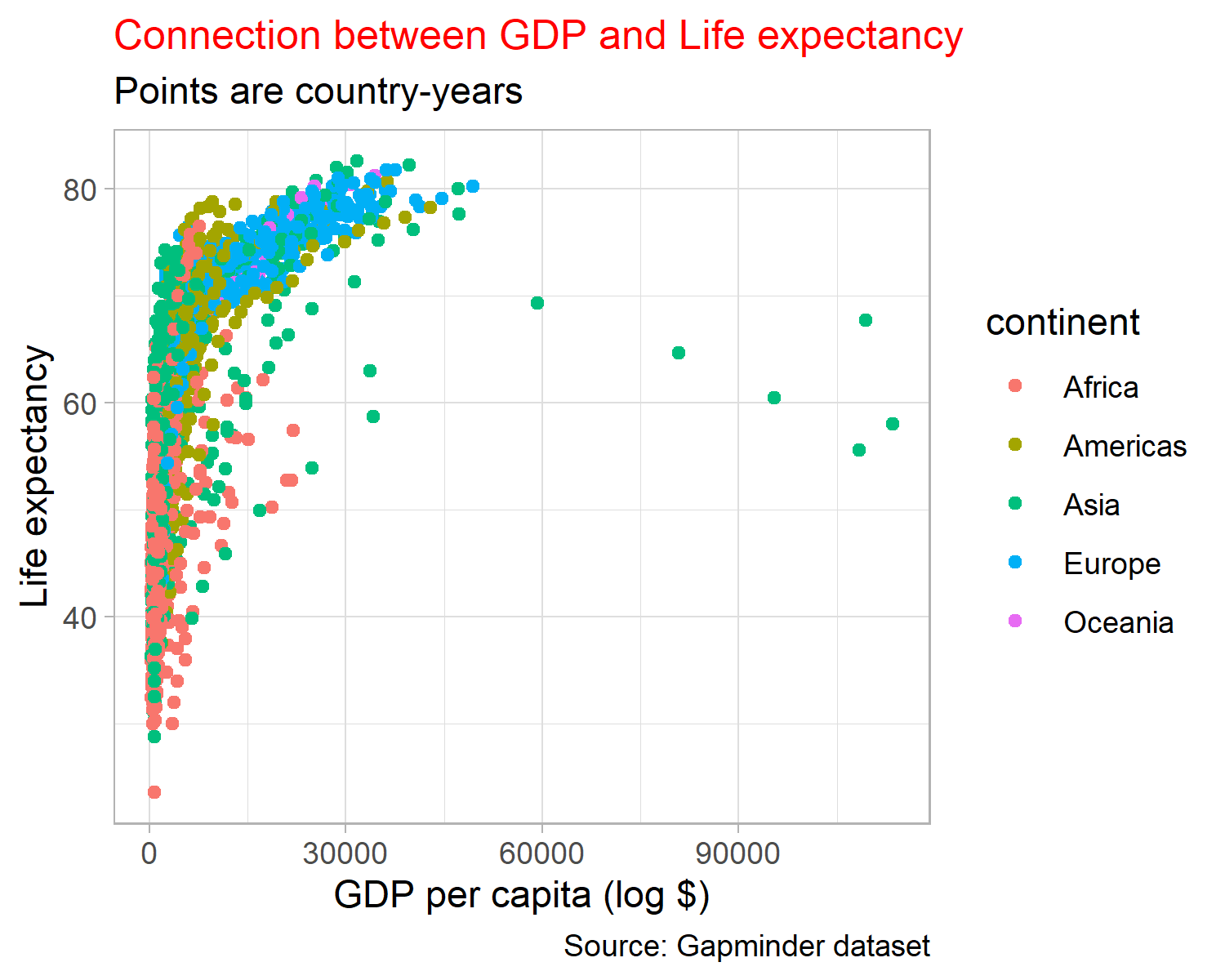
Az fenti script kibővítésével az egyes kontinensek adatait különböző színnel ábrázolhatjuk, az x és y tengelyt elnevezhetjük, a histogramnak címet és alcímet adhatunk, illetve az adataink forrását is feltüntethetjük az alábbi módon:

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"  
 )



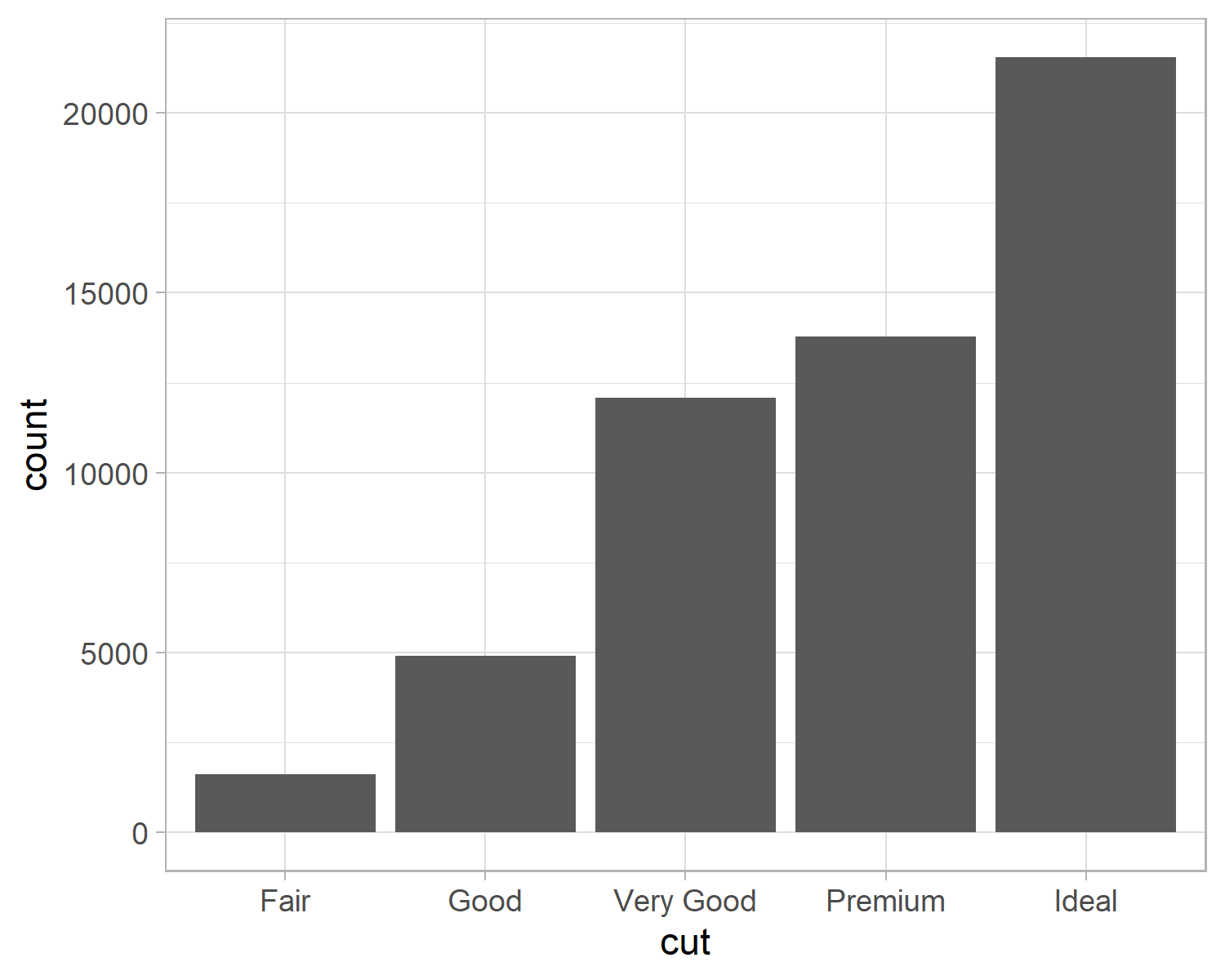
Az ábrán található feliratok méretének, betűtípusának és betűszínének megválasztásra is lehetőségünk van.

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"  
 ))



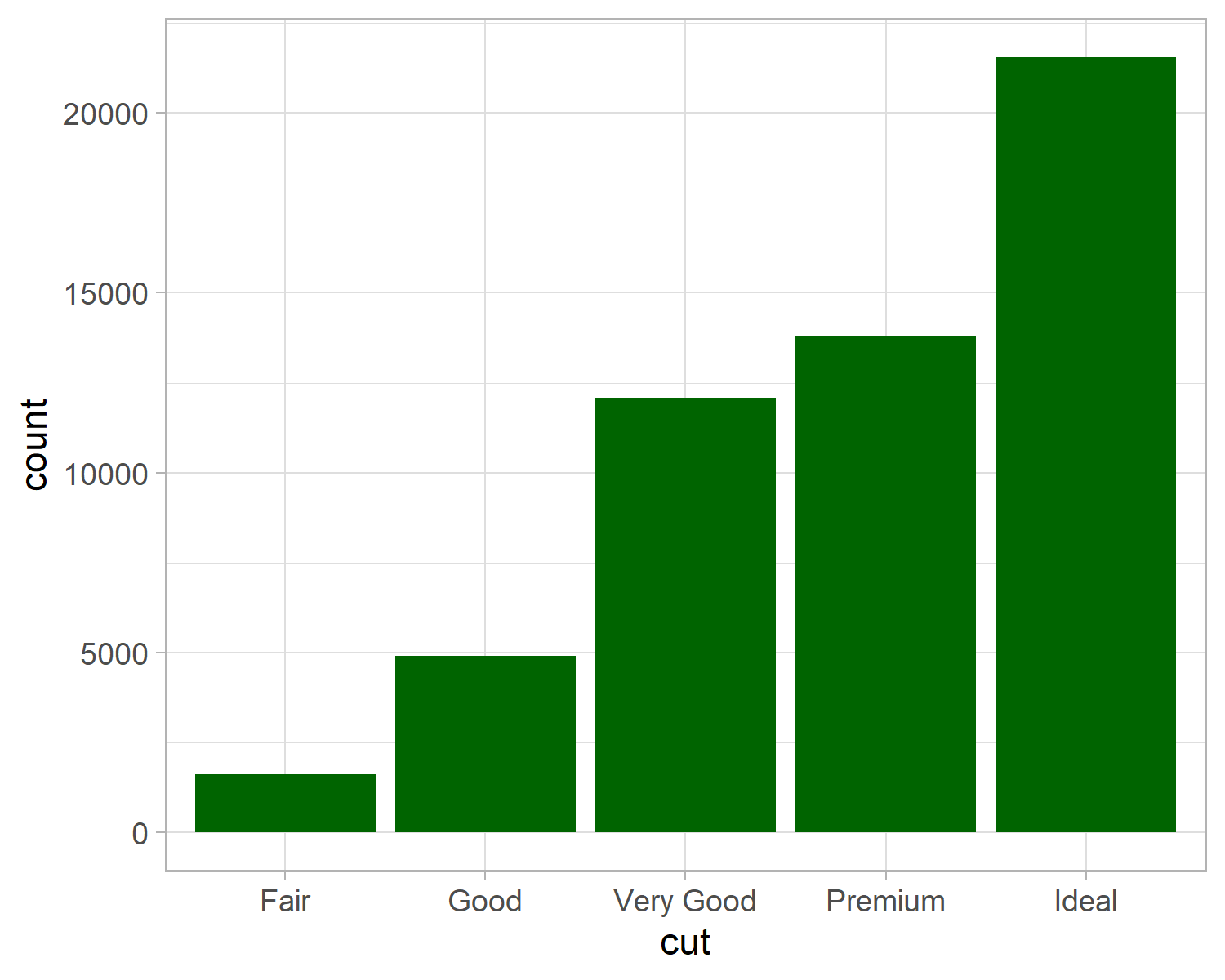
Készíthetünk oszlopdiagramot is, amit a ggplot2 diamonds adatkészletén személtetünk

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



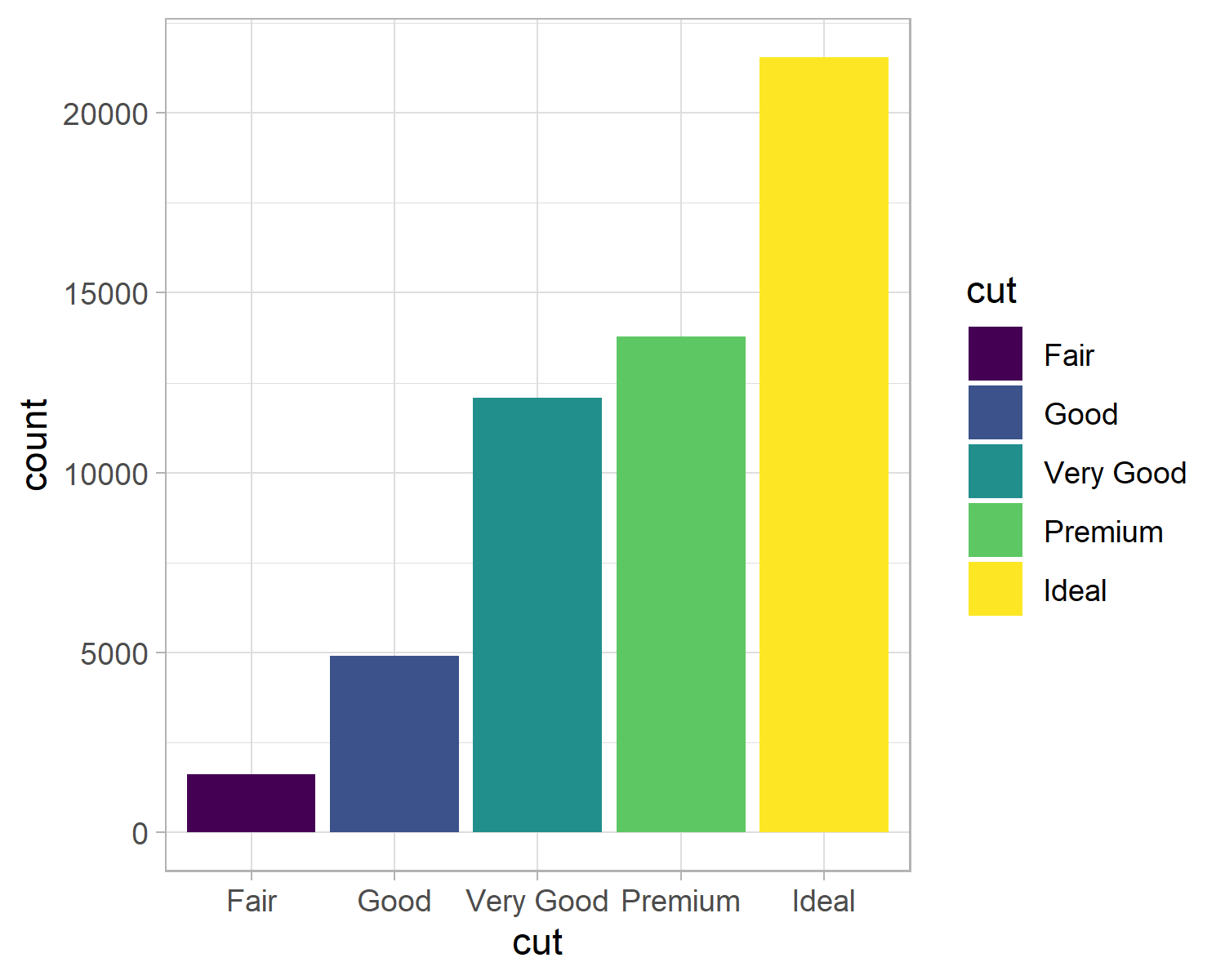
Itt is lehetőségünk van arra, hogy a diagram színét megváltoztassuk.

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

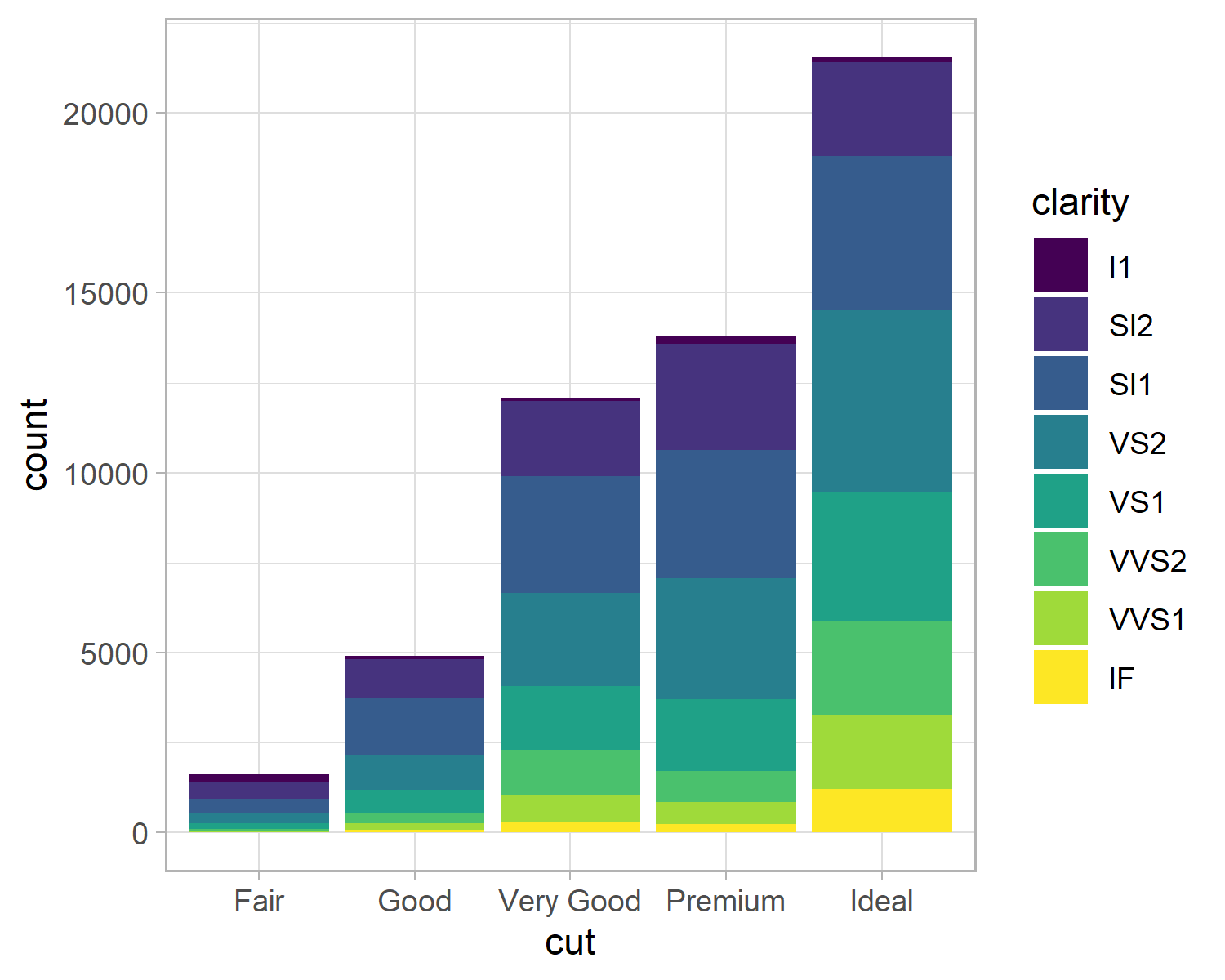


De arra is lehetőségünk van, hogy az egyes oszlopok eltérő színűek legyenek.

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, 391402.

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

Burtejin, Zorgit. 2016. “Csoportosítás (Klaszterezés).” In, 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): 17751781.

Deveaud, Romain, Eric SanJuan, and Patrice Bellot. 2014. “Accurate and Effective Latent Concept Modeling for Ad Hoc Information Retrieval.” *Document Numérique* 17 (1): 6184.

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. 2013a. “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.” *Political Analysis* 21 (3): 267297.

———. 2013b. “Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts.” *Political Analysis* 21 (3): 267297.

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.

Laver, Michael, Kenneth Benoit, and John Garry. 2003. “Extracting Policy Positions from Political Texts Using Words as Data.” *American Political Science Review*, 311331.

Laver, Michael, and John Garry. 2000. “Estimating Policy Positions from Political Texts.” *American Journal of Political Science*, 619634.

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*.

Mikolov, Tomas, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. “Advances in Pre-Training Distributed Word Representations.” In.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. “Glove: Global Vectors for Word Representation.” In, 15321543.

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): 10641082.

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

Spirling, Arthur, and Pedro L Rodriguez. n.d.a. “Word Embeddings.” <https://polmeth.mit.edu/sites/default/files/documents/Pedro_Rodriguez.pdf>.

———. n.d.b. “Word Embeddings.” <https://polmeth.mit.edu/sites/default/files/documents/Pedro_Rodriguez.pdf>.

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

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

———. 2007b. *Szövegbányászat*. Budapest: Typotext.

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): 205231.

1. A *tidyverse* megközelítés miatt a kötetben szereplő R kód követi a “The tidyverse style guide” dokumentációt (<https://style.tidyverse.org/>) [↑](#footnote-ref-25)
2. Az RStudio-ban a pipe operátor billentyű kombinációja a Ctrl + Shift + M [↑](#footnote-ref-27)
3. Köszönjük Andrew Heissnek a kitűnő példát. [↑](#footnote-ref-28)
4. A könyv terjedelme miatt ezt a témát itt csak bemutatni tudjuk, de minden részletre kiterjedően nem tudunk elmélyülni benne. Kíváló 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-35)
5. Az itt használt kódok az alábbiakon alapulnak: <http://www.akosmate.com/QTA_SZISZ_2019/week03_descriptives_i/session3_r_script.html>, <https://rdrr.io/cran/quanteda/man/dfm_weight.html>, <https://rdrr.io/cran/quanteda/man/dfm_tfidf.html> [↑](#footnote-ref-51)
6. A mintába nem került be Rogán Antal, akinek csak egy darab napirend előtti felszólalása volt. [↑](#footnote-ref-59)
7. 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-60)
8. Azért nem a Vona Gábor beszédét választottuk, mert az gyaníthatóan egy kiugró érték ami nem reprezentálja a sokaságot megfelően. [↑](#footnote-ref-63)
9. ([**schütze2008a:294-307?**](#ref-schütze2008a:294-307)) [↑](#footnote-ref-67)
10. 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-loughran2011) ([2011](#ref-loughran2011)); [Máté, Sebők, and Barczikay](#ref-máté2021) ([2021](#ref-máté2021)) [↑](#footnote-ref-74)
11. 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-75)
12. A szótár elérhető a könyv online verziójának a GitHub repozitorijából. (LINK) [↑](#footnote-ref-78)
13. A csoportosított adatokkal való munka bővebb bemutatását lsd. a Függelékben [↑](#footnote-ref-79)
14. A témával részletesebben is foglalkoztunk a [Máté, Sebők, and Barczikay](#ref-máté2021) ([2021](#ref-máté2021)) 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-82)
15. 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-90)
16. A teljes törvényeket és a metadatokat tartalmazó adatbázisokat a <https://cap.tk.hu/> honlapról lehet letölteni. [↑](#footnote-ref-94)
17. A ldatuning R csomagban további indikátor implementációja található, ami a perplexityhez hasonlóan minimalizásra alapoz ([Arun et al.](#ref-arun2010finding) ([2010](#ref-arun2010finding)), [Cao et al.](#ref-cao2009density) ([2009](#ref-cao2009density))), illetve maximalizálásra ([Deveaud, SanJuan, and Bellot](#ref-deveaud2014accurate) ([2014](#ref-deveaud2014accurate)), [Griffiths and Steyvers](#ref-griffiths2004) ([2004](#ref-griffiths2004))) [↑](#footnote-ref-100)
18. a tidy formátumról bővebben: <https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html> [↑](#footnote-ref-101)
19. Az stm csomaghoz tartozó leírás: <https://cran.r-project.org/web/packages/stm/vignettes/stmVignette.pdf> [↑](#footnote-ref-111)
20. Egy kíváló tanulmányban [Spirling and Rodriguez](#ref-spirlingword) ([n.d.a](#ref-spirlingword)) összehasonlítják a Word2Vec és GloVe módszereket, különbözö paraméterekkel, adatbázisokkal. Amennyiben valakit komolyabban érdekelnek a szóbeágyazás gyakorlati alkalmazásának a részletei annak mindenképp ajánljuk elolvasásra. [↑](#footnote-ref-117)
21. A Magyar CAP Projekt által kezelt adatbázisok itt megtalálhatóak: <https://cap.tk.hu/adatbazisok> [↑](#footnote-ref-118)
22. A futtatásra használt PC nem különösebben erős: 4 magos Intel Core i5-4460 (3.2GHz) CPU és 16GB RAM [↑](#footnote-ref-120)
23. A mintába nem került be Rogán Antal, akinek csak egy darab napirend előtti felszólalása volt. [↑](#footnote-ref-124)
24. 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-125)
25. Azért nem a Vona Gábor beszédét választottuk, mert az gyaníthatóan egy kiugró érték ami nem reprezentálja a sokaságot megfelően. [↑](#footnote-ref-130)