Universität Hamburg
Fachbereich: Sozialwissenschaften
Fachgebiet: Politikwissenschaft
Seminar: Musterseminar
Dozenten: Prof. Dr. Kai-Uwe Schnapp
PD Dr. Falk Daviter
Wintersemester 2018/19

Forschungsarbeit

Geschlechterunterschiede im Deutschen Bundestag

Arbeitstitel

tba

Josef Holnburger
Matrikelnummer: XXX
XXX
XXX
E-Mail: josef@holnburger.com

Inhaltsverzeichnis

1 Einleitung

Zusammen mit Gina-Gabriela Görner analysiere ich die Protokolle des Deutschen Bundestages auf mögliche Geschlechterunterschiede. Hierfür wollen wir die Anzahl aber auch Inhalte der Reden mehrer Wahlperioden des Bundestags untersuchen. Dieses Projekt wurde auch für die 4. International Conference on Computational Social Science eingereicht und wir dürfen es dort mit einem Plakat vorstellen.

Wir orientieren uns vor allem an der Forschung von Bäck et al. (2014), welche das schwedische Parlament auf mögliche Geschlechterunterschiede und Diskriminierung hin untersucht haben. In dieser Studie wurden Unterschiede sowohl in der Anzahl als auch bezüglich des Inhalts der Reden festgestellt. Auuch im schwedischen Parlament sind Männer deutlich häufiger zu hören – obwohl es mit einem Frauenanteil von 40 Prozent die höchste Quote europäischer Parlamente aufweist (Bäck et al. 2014: 505). Männer sprechen in ihren Reden häufiger über *hard topics*, bei *soft topics* ist der Redeanteil hingegen ausgeglichen (ebd: 513ff.). Die Konstruktion der *hard* und *soft topics* geht dabei auf Wangnerud (2000) zurück und ist nicht unkritisch – hier werden durchaus Geschlechterstereotype aufrechterhalten oder gar reproduziert, indem "typische" Frauen und Männerthemen identifiziert werden. Wangnerud hat ihn ihrer Untersuchung die Mitglieder des schwedischen *Riksdag* bezüglich ihrer Aktivitäten befragt. Das Ergebnis von Bäck et al. (2014) ist deshalb auch nicht besonders überraschend – bestätigt es doch nur, dass die Fachpolitiker_innen häufiger über ihre Themen auch im Plenum reden.

In dieser Arbeit soll anders vorgegangen werden. Die Inhalte der Reden im Bundestaug sollen ohne voherige Identifikation vermeintlicher Frauen- und Männerthemen untersucht werden. Hierbei nutzen wir die Möglichkeiten des *Topic Modelling* um zunächst generell Themen der Reden im Bundestags zu identifizieren und diese anschließend auf mögliche Geschlechterunterschiede untersuchen. Natürlich wollen auch wir die Unterschiedlichen Redeanteile untersuchen.

Da ich den Prinzipien der Open Science sehr viel abgewinnen kann, soll die Erhebung und Auswertung möglichst transparent und nachvollziehbar dargestellt werden.

2 Datenerhebung Teil 1

Die Reden im Deutschen Bundestag sind in den Protokollen dokumentiert und lassen sich online abrufen. Praktischerweise liegen die Daten seit der aktuellen Wahlerperiode auch im TEI-Format (Text Encoding Initiative) vor. Dies erleichtert die Analyse der Protkolle erheblich. Die Datenerhebung und Auswertung erfolgt mit der Programmiersprache R (R Core Team 2018) und der `tidyverse` Packetsammlung (Wickham 2017).

Die Datenerhebung soll zunächst an einem Beispielprotokoll gezeigt werden – wir nutzen hierfür das Protokoll der 61. Sitzung des 19. Bundestages. Das Protokoll liegt dabei sowohl als PDF, als TXT und auch als XML-Datei vor. Letzeres wird für diese Arbeit herangezogen.

Mit dem Packet `xml2` (Wickham et al. 2018) kann das Protokoll ausgelesen und anschließend in ein passendes Format umgewandelt werden. Mit der Funktion `read_html()` wird das vollständige Protokoll in der Variable `prot_file` eingelesen. Die Umwandlung der einzelnen Knoten und Attribute des XML-Dokuments erfolgt mit dem `rvest` Packet (Wickham 2016).

Da für diese Auswertung nur die Reden im Deutschen Bundestag herangezogen werden (und angehängte Dokumente sowie Anwesenheitslisten irrelevant sind), soll nur ein Teil des Protkolls untersucht werden. Mittels der Funktion `xml_find_all(//rede")` können alle Einträge unter dem Knoten “rede” herausgefiltert werden.

```
library(tidyverse)
library(xml2)
library(rvest)


prot_file <- read_html("https://www.bundestag.de/blob/577958/b2d1fce9b7dec32a1403a2ec

prot_overview <- prot_file %>%
  xml_find_all("//rede")
```

Die Datei soll anschließend in einen *Dataframe* umgewandelt werden. Dies erleichtert die weitere Arbeit und im weiteren Verlauf können die Daten einfacher nach nach Geschlecht, Partei, Datum oder Wahlperiode gefiltert werden. Hierbei wird vor allem mit den Funktionen `xml_node()` und `xml_attr()` gearbeitet. Zum Verständnis bietet sich hier ein kleiner Diskurs an.

2.1 XML-Knoten und Attribute

Nachdem die Datei eingelesen wurde, lohnt sich ein Blick auf die Rohdaten:

```
<rede id="ID196105400">
  <p klasse="redner"><redner id="11004826"><name><vorname>Siemtje</vorname><nachname
  <p klasse="J_1">Lieber Jürgen Trittin, Sie haben gerade eigentlich das wiederholt,
  <kommentar>(Beifall bei der SPD)</kommentar>
</rede>
```

In diesem Beispiel wird das XML-Fragment in die Variable `xml_example` geladen und ausgewertet. Die Knoten eines XML-Documents werden durch `<>` und `</>` eingefasst. Beispielsweise können die Knoten mit den Namen “Kommentar” folgendermaßen extrahiert werden:

```
xml_example %>% xml_nodes("kommentar")
```

```
## {xml_nodeset (1)}
## [1] <kommentar>(Beifall bei der SPD)</kommentar>
```

Bei der Ausgabe fällt jedoch auf, dass die Datei weiterhin eine XML-Datei bleibt und die Knoteninformationen ebenfalls extrahiert werden. Mittels der Funktion `xml_text()` kann das Ergebniss in einen in einen Character-String umwandelt werden.

```
xml_example %>% xml_nodes("kommentar") %>% xml_text()
```

```
## [1] "(Beifall bei der SPD)"
```

Die Ergebnisse werden in einer Liste zusammengefasst und können beispielsweise in einem

Datenframe umgewandelt werden.

Die Attribute eines Knotens finden sich in den Klammern nach dem Gleichheitszeichen: `<knotenname attribut=inhalt">...`. Die Werte eines Attributes (und auch den Attributnamen) können mit der Funktion `xml_attr()` extrahiert werden.

```
xml_example %>% xml_node("rede") %>% xml_attr("id")
```

```
## [1] "ID196105400"
```

Mit dieser kurzen Exkursion können wir nun eine Funktion bauen, welche auf die für uns relevanten Daten aus dem XML-Dokument extrahiert und anschließend in einen Datenframe umwandelt.

```
get_overview_df <- function(x){
  rede_id <- x %>% xml_attr("id")
  redner_id <- x %>% xml_node("redner") %>% xml_attr("id")
  redner_vorname <- x %>% xml_node("redner") %>% xml_node("vorname") %>% xml_text()
  redner_nachname <- x %>% xml_node("redner") %>% xml_node("nachname") %>% xml_text(
  redner_fraktion <- x %>% xml_node("redner") %>% xml_node("fraktion") %>% xml_text(
  redner_rolle <- x %>% xml_node("rolle_kurz") %>% xml_text()

  data_frame(rede_id, redner_id, redner_vorname, redner_nachname, redner_fraktion, re
}
```

Wir können mit dieser Funktion nun die vorher eingelesen XML-Datei in einen Datenframe umwandeln und auswerten.

```
overview_df <- get_overview_df(prot_overview)


overview_df
```

```
## # A tibble: 144 x 6
##    rede_id redner_id redner_vorname redner_nachname redner_fraktion
##    <chr>   <chr>     <chr>          <chr>           <chr>
```

```
##  1 ID1961~ 11003196  Andrea         Nahles           SPD
##  2 ID1961~ 11004873  Ulrike         Schielke-Ziesi~ AfD
##  3 ID1961~ 11002666  Hermann        Gröhe            CDU/CSU
##  4 ID1961~ 11004179  Johannes       Vogel            FDP
##  5 ID1961~ 11004012  Matthias W.    Birkwald         DIE LINKE
##  6 ID1961~ 11003578  Markus         Kurth            BÜNDNIS 90/DIE~
##  7 ID1961~ 11003142  Hubertus       Heil             <NA>
##  8 ID1961~ 11004856  Jürgen         Pohl             AfD
##  9 ID1961~ 11002812  Max            Straubinger      CDU/CSU
## 10 ID1961~ 11004941  Gyde           Jensen           FDP
## # ... with 134 more rows, and 1 more variable: redner_rolle <chr>
```

2.2 Zwischenfazit

Wir konnten mit wenigen Zeilen Code das XML-Format in einen Datenframe umwandeln, welcher uns die weitere Arbeit erheblich erleichter. So könnten wir sehr schnell sagen, wie viele Reden es von den einzelnen Fraktionen zur 61. Sitzung des 19. Bundestags gab:

```
overview_df %>%
  group_by(redner_fraktion) %>%
  summarise(reden = n()) %>%
  arrange(-reden)
```

```
## # A tibble: 8 x 2
##   redner_fraktion       reden
##   <chr>                 <int>
## 1 CDU/CSU                  41
## 2 SPD                      27
## 3 AfD                      20
## 4 BÜNDNIS 90/DIE GRÜNEN    16
## 5 FDP                      16
```

```
## 6 DIE LINKE                15
## 7 <NA>                      6
## 8 fraktionslos              3
```

Da *NA* Fraktionen sind dabei die Reden von Ministern und Gästen. Sie werden keiner Fraktion zugeordnet. Insgesamt gab es 144 Reden an diesem Tag.

Uns interessieren natürlich nun nicht nur die Anzahl der Reden, sondern auch deren Inhalt. Wir untersuchen hierfür alle Knoten eine Ebene unter den "rede"-Knoten.

```
prot_speeches <- prot_file %>%
  xml_find_all("//rede/*")
```

Wir bauen wieder eine Funktion, um alle Inhalte der Reden zu extrahieren. Diese Funktion ist ein wenig komplexer, da sie unter anderem die Funktion `map()` aus dem `purrr` Packet nutzt (ebenfalls `tidyverse`) – für weitere Informationen über die Funktion `map()` bietet sich dieses Tutorial an.

Außerdem müssen die Rohdaten etwas angepasst werden, da die Aussagen des Präsidiums sonst falsch zugeordnet werden.

```
get_speeches_df <- function(x){
  raw <- x
  rede <- x %>% xml_text()
  id <- x %>% xml_node("redner") %>% xml_attr("id")
  vorname <- x %>% xml_node("vorname") %>% xml_text()
  nachname <- x %>% xml_node("nachname") %>% xml_text()
  fraktion <- x %>% xml_node("fraktion") %>% xml_text()
  rolle <- x %>% xml_node("rolle_kurz") %>% xml_text()
  typ <- x %>% xml_name()
  status <- x %>% xml_attr("klasse")

  data_frame(raw, rede, id, vorname, nachname, fraktion, rolle, typ, status) %>%
    mutate(rede_id = map(raw, ~xml_parent(.) %>% xml_attr("id")) %>% as.character())
```

```
    select(-raw) %>%
    mutate(status = ifelse(typ == "kommentar", typ, status)) %>%
    mutate(status = ifelse(typ == "name", "präsidium", status)) %>%
    mutate(fraktion = case_when(
      typ == "name"        ~ "präsidium",
      !is.na(rolle)        ~ "andere",
      TRUE                 ~ fraktion)) %>%
    fill(id, vorname, nachname, fraktion) %>%
    mutate(präsidium = ifelse(fraktion == "präsidium", TRUE, FALSE)) %>%
    mutate(fraktion = ifelse(fraktion == "präsidium", NA, fraktion)) %>%
    filter(!status %in% c("T_NaS", "T_Beratung", "T_fett", "redner")) %>%
    filter(!typ %in% c("a", "fussnote", "sup")) %>%
    select(rede_id, rede, id, vorname, nachname, fraktion, präsidium, typ, status)
}

get_overview_df <- function(x){
  rede_id <- x %>% xml_attr("id")
  redner_id <- x %>% xml_node("redner") %>% xml_attr("id")
  redner_vorname <- x %>% xml_node("redner") %>% xml_node("vorname") %>% xml_text()
  redner_nachname <- x %>% xml_node("redner") %>% xml_node("nachname") %>% xml_text(
  redner_fraktion <- x %>% xml_node("redner") %>% xml_node("fraktion") %>% xml_text(
  redner_rolle <- x %>% xml_node("rolle_kurz") %>% xml_text()
  sitzung <- x %>% xml_find_first("//sitzungsnr") %>% xml_text() %>% as.integer()
  datum <- x %>% xml_find_first("//datum") %>% xml_attr("date") %>% lubridate::dmy()
  wahlperiode <- x %>% xml_find_first("//wahlperiode") %>% xml_text() %>% as.integer

  data_frame(rede_id, redner_id, redner_vorname, redner_nachname, redner_fraktion, re
}

speeches_df <- get_speeches_df(prot_speeches)
```

Mittels dieses Datenframes ist es nun möglich, nur die Aussagen von beispielsweise Andrea Nahles zu untersuchen – ohne Unterbrechungen und Fragen von anderen Abgeordneten "mitzuschneiden" oder Aussagen des Präsidiums mitzunehmen.

Hier ein Beispiel:

```
speeches_df %>%
  filter(typ != "kommentar") %>%
  filter(präsidium == FALSE) %>%
  filter(id == "11003196") %>%
  pull(rede) %>%
  cat(fill = TRUE)
```

```
## Herr Präsident! Meine lieben Kolleginnen und Kollegen! Auch in dieser Woche verabs
## Mit der heutigen Rentenreform vollziehen wir einen grundsätzlichen Richtungswechse
## Wir sichern damit ein Rentenniveau auf dem heutigen Level. Das ist wirklich eine s
wenn sie eben ergänzend gedacht ist, nicht ersetzend. Das ist der entscheidende Punkt
## Denn die gesetzliche Rentenversicherung ist und bleibt die zentrale Säule im deuts
## Die Rentenreform folgt einem einfachen Prinzip: Wer ein Leben lang arbeitet, der v
## Ich betone: Ich benutze den Begriff „verdient" bewusst. Denn die Rente ist kein Al
## Uns ist die Stärkung der umlagefinanzierten Rente ja auch deswegen so wichtig, wei
## Im Gegensatz zu den privaten steht die gesetzliche Rente blendend da. Würde man au
## Die umlagefinanzierte Rente ist deswegen der kapitalgedeckten überlegen.
## Ich spreche jetzt in diesem Hohen Haus auch etwas aus, was vielleicht nicht alle g
## oder wir lassen zu, dass die Renten immer weiter sinken und entwertet werden.
## Wenn wir das aber zulassen, muss die junge Generation einem solchen System irgendw
## Deswegen ist aus meiner Sicht die Sicherung des Rentenniveaus in diesem System auc
## Jetzt sagen manche, das sei nicht finanzierbar. Das ist ein ziemlich scheinheilige
## Denn niemand wird ja wohl bestreiten, dass das Geld für eine auskömmliche Rente im
## Das, was wir heute beschließen, ist finanziert. Bis 2025 ist das Rentenniveau klar
## Wir steigen darüber hinaus in die Bildung einer Demografierücklage ein.
## Damit schaffen wir die Voraussetzung, um den Steueranteil zur Finanzierung der Ren
## Das wird wahrscheinlich auch der Weg der Zukunft sein. Darüber wird aber in der Re
```

```
## Wenn es aber etwas gibt, was wir klären müssen, dann ist das doch die Frage: Wolle
## Einen Weg zur Finanzierung werden wir in einem reichen Land wie Deutschland sicher
## Letzter Satz. Wenn es also einen Gradmesser für die soziale Sicherheit in Deutschl
## Vielen Dank.
```

Somit könnten wir für dieses Protokoll die einzelnen Reden (aber zum Beispiel auch Zwischenfragen) von Abgeordneten gezielt auf deren Inhalte untersuchen. Wir können noch nicht die Zwischenrufe und den Applaus nach Abgeordneten bzw. Fraktionen auswerten. Dies wäre mit sogenannten *regular experesions* aber möglich.

Wie wir alle aktuellen Protokolle auswerten, behandeln wir in Daten wir in Kapitel 3. Die beiden Funktionen speichern wir im Ordner “functions”.

3 Datenerhebung Teil 2

Leider gibt es keine Möglichkeit, die XML-Protokolle des Deutschen Bundestages gesammelt herunterzuladen. Zwar finden sich auf *Open Data*-Seite des Bundestags[1] die Verweise auf die Bundestagsprotkolle, allerdings können diese nur umständlich einezln heruntergeladen werden. Durch das Auslesen der Netzwerkdaten nach einem Klick auf die nächsten fünf Protkolle eine Webseite gefunden werden, auf welcher jeweils fünf Protkolle gespeichert sind[2]. Durch setzen des `offset=0` auf `5, 10, 15, 20, …` können wir die weiteren Protokolle abrufen.

Mittels `rvest` und `xml2` ziehen wir uns zunächst die Nummer des letzten Bundestagsprotokolls. Über die Funktion `seq()` und `paste0()` können wir anschließend alle für den weiteren Verlauf notwendigen URLs erstellen.

```
bt_website <- "https://www.bundestag.de/ajax/filterlist/de/service/opendata/-/543410"

last_protocol <- bt_website %>%
  read_html() %>%
  xml_find_first("//strong") %>%
  xml_text(trim = TRUE) %>%
  str_extract("\\d+")

prot_websites <- paste0(bt_website, "?offset=", seq(0, last_protocol, 5))
```

Insgesamt müssen wir also 13 Webseiten aufrufen und uns jeweils fünf Protokolle herunterladen.

Der Aufwand ist hier noch überschaubar. Natürlich laden wir die Protokolle dennoch nicht per Hand herunter, sondern erstellen uns hierfür ein kleines Script. Da die meisten Rechner auch mit mehr als nur einem Prozessor ausgestattet sind, können wir die Funktion auch auf mehreren Prozessoren ausführen. Wir sprechen hier von *multiprocessing*. Dies

[1] https://www.bundestag.de/service/opendata
[2] https://www.bundestag.de/ajax/filterlist/de/service/opendata/-/543410?offset=0

realisieren wir über das Packet `furrr` (Vaughan/Dancho 2018) - eine Abwandlung des bereits genutzten `purrr`.

Wir schreiben uns zunächst eine Funktion, um die Links der Webseiten zu extrahieren.

```
get_prot_links <- function(x){
  x %>%
    read_html() %>%
    html_nodes(".bt-link-dokument") %>%
    html_attr("href") %>%
    paste0("https://www.bundestag.de", .)
  }

get_prot_links(bt_website)
```

```
## [1] "https://www.bundestag.de/blob/578466/7430bccaf792e7bc55e84d5e64675820/19062-d
## [2] "https://www.bundestag.de/blob/577958/e2063c0f51a32690a269f48aa6102c1d/19061-d
## [3] "https://www.bundestag.de/blob/577622/da97888b713abb16ed2070836504b83a/19060-d
## [4] "https://www.bundestag.de/blob/575138/b5395a975d1c55838da0e52251018160/19059-d
## [5] "https://www.bundestag.de/blob/574826/0e3659e11c1c3cdbfa621369cd16735a/19058-d
```

Dies wenden wir nun auf alle 13 Webseiten an und speichern die Dateien anschließend im Ordner `data\protokolle`.

```
library(furrr)
plan(multiprocess)

prot_links <- future_map(prot_websites, ~get_prot_links(.)) %>% unlist()

prot_links %>% future_map(~download.file(., file.path("data", basename(.))))
```

Wir waren erfolgreich und konnten in wenigsten Sekunden alle aktuellen Protokolle herunterladen. Wir können sie jetzt auslesen und dabei auch unsere bereits erstellten Funktionen verwenden. Jetzt können wir alle Dateien einlesen, die Übersicht der Reden oder

den Inhalt der Reden extrahieren und anschließend auswerten.

```
source("functions/get_overview_df.R")
source("functions/get_speeches_df.R")
```

```
prot_files <- list.files("data", full.names = TRUE)

prot_extract <- map(prot_files, ~read_html(.) %>% xml_find_all("//rede"))

class(prot_extract) <- "xml_nodeset"

prot_overview <- map_dfr(prot_extract, ~get_overview_df(.))
```

3.1 Zwischenfazit

Wir konnten nun alle Dateien herunterladen und anschließend alle Protkolle in R einlesen. Mit unserer Funktion `get_overview_df()` konnten wir alle Protokolle in einen für uns passenden Datenframe umwandeln. Insgesamt können wir derzeit 5.777 Reden des aktuellen Bundestags untersuchen – etwa nach der Person, welche die meisten reden gehalten hat.

```
prot_overview %>%
  group_by(redner_id, redner_vorname, redner_nachname,
           redner_fraktion, redner_rolle) %>%
  summarise(reden = n()) %>%
  arrange(-reden)
```

```
## # A tibble: 773 x 6
## # Groups:   redner_id, redner_vorname, redner_nachname, redner_fraktion
## #   [728]
##    redner_id redner_vorname redner_nachname redner_fraktion redner_rolle
##    <chr>     <chr>          <chr>           <chr>           <chr>
```

```
##  1 11002617  Peter          Altmaier       <NA>           Bundesminis~
##  2 999990073 Olaf           Scholz         <NA>           Bundesminis~
##  3 11004427  Volker         Ullrich        CDU/CSU        <NA>
##  4 11001478  Angela         Merkel         <NA>           Bundeskanzl~
##  5 11004809  Heiko          Maas           <NA>           Bundesminis~
##  6 11004851  Frauke         Petry          fraktionslos   <NA>
##  7 11004798  Alexander Graf Lambsdorff     FDP            <NA>
##  8 11003625  Andreas        Scheuer        <NA>           Bundesminis~
##  9 999990074 Svenja         Schulze        <NA>           Bundesminis~
## 10 11003638  Jens           Spahn          <NA>           Bundesminis~
## # ... with 763 more rows, and 1 more variable: reden <int>
```

Die Minister_innen haben am häuiigsten im Bundestag geredet, der MdB Volker Ullrich landet erst af Platz 4 mit insgesamt vierzig Reden in der aktuellen Wahlperiode. Überraschenderweise hat Frauke Petry sehr viele Reden gehalten: 35 Stück. Das sind deutlich mehr als in ihrer damaligen Zeit im Landesparlament.

Literatur- und Quellverzeichnis

Bäck, Hanna/Debus, Marc/Müller, Jochen (2014): Who Takes the Parliamentary Floor? The Role of Gender in Speech-Making in the Swedish "Riksdag". In: *Political Research Quarterly*, 67 (3), 504–518.

R Core Team (2018): R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Text abrufbar unter: https://www.R-project.org/.

Vaughan, Davis/Dancho, Matt (2018): Furrr: Apply Mapping Functions in Parallel Using Futures. Text abrufbar unter: https://CRAN.R-project.org/package=furrr.

Wangnerud, Lena (2000): Testing the Politics of Presence: Women's Representation in the Swedish Riksdag. In: *Scandinavian Political Studies*, 23 (1), 67–91.

Wickham, Hadley (2016): Rvest: Easily Harvest (Scrape) Web Pages. Text abrufbar unter: https://CRAN.R-project.org/package=rvest.

Wickham, Hadley (2017): Tidyverse: Easily Install and Load the 'Tidyverse'. Text abrufbar unter: https://CRAN.R-project.org/package=tidyverse.

Wickham, Hadley/Hester, James/Ooms, Jeroen (2018): Xml2: Parse XML. Text abrufbar unter: https://CRAN.R-project.org/package=xml2.