

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

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Datenerhebung Teil 1</b>	<b>2</b>
2.1	XML-Knoten und Attribute . . . . .	3
2.2	Zwischenfazit . . . . .	5
<b>3</b>	<b>Datenerhebung Teil 2</b>	<b>9</b>
	<b>Literatur- und Quellverzeichnis</b>	<b>10</b>

# 1 Einleitung

Zusammen mit Gina-Gabriele Görner analysiere ich die Protokolle des Deutschen Bundestages auf mögliche Geschlechterunterschiede. Hierfür wollen wir die Anzahl aber auch Inhalte der Reden mehrerer 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. Auch 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 in 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 Bundestag sollen ohne vorherige Identifikation vermeintlicher Frauen- und Männerthemen untersucht werden. Hierbei nutzen wir die Möglichkeiten des *Topic Modelling* um zunächst generell Themen der Reden im Bundestag 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 Protokolle 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. Letzteres wird für diese Arbeit herangezogen.

Mit dem Paket `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` Paket (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 Protokolls 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/b2d1fce9b7dec32a1403a2e0")

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 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, redner_rolle)
}
```

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 erleichtert. 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` Paket 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)
}

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 Richtungswechsel
## 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 deutschen Sozialversicherungssystem.

## Die Rentenreform folgt einem einfachen Prinzip: Wer ein Leben lang arbeitet, der verdient eine Rente.

## Ich betone: Ich benutze den Begriff „verdient“ bewusst. Denn die Rente ist kein Almosen.

## Uns ist die Stärkung der umlagefinanzierten Rente ja auch deswegen so wichtig, weil sie die Grundlage für die soziale Sicherheit bildet.

## Im Gegensatz zu den privaten steht die gesetzliche Rente blendend da. Würde man auf sie verzichten, wäre das ein katastrophaler Fehler.

## Die umlagefinanzierte Rente ist deswegen der kapitalgedeckten überlegen.

## Ich spreche jetzt in diesem Hohen Haus auch etwas aus, was vielleicht nicht alle ganz verstehen werden.

## 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 irgendwann zustimmen.

## Deswegen ist aus meiner Sicht die Sicherung des Rentenniveaus in diesem System auch eine politische Aufgabe.

## Jetzt sagen manche, das sei nicht finanzierbar. Das ist ein ziemlich scheinhaftes Argument.

## Denn niemand wird ja wohl bestreiten, dass das Geld für eine auskömmliche Rente im Sozialversicherungssystem vorhanden ist.

## Das, was wir heute beschließen, ist finanziert. Bis 2025 ist das Rentenniveau klar gesichert.

## Wir steigen darüber hinaus in die Bildung einer Demografierücklage ein.

## Damit schaffen wir die Voraussetzung, um den Steueranteil zur Finanzierung der Rente zu senken.

## Das wird wahrscheinlich auch der Weg der Zukunft sein. Darüber wird aber in der Zukunft entschieden werden.

## Wenn es aber etwas gibt, was wir klären müssen, dann ist das doch die Frage: Woher soll das Geld kommen?

## Einen Weg zur Finanzierung werden wir in einem reichen Land wie Deutschland sicher finden.

## Letzter Satz. Wenn es also einen Gradmesser für die soziale Sicherheit in Deutschland gibt, dann ist das die gesetzliche Rente.

## 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 expressions* 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 Bundesatages gesammelt herunterzuladen. Zwar findet sich auf *Open Data*-Seite des Bundestags<sup>1</sup> ein Verweis auf die Protokolle im XML-Format, allerdings lassen sie sich von dieser Seite nur mit mehreren Klicks installieren.\$

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

---

<sup>1</sup><https://www.bundestag.de/service/opendata>

# 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/>.

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>.