KED2022 – Miniprojekt

Dario Haab, Valentina Meyer, Nils Brun

Einleitung

Thema

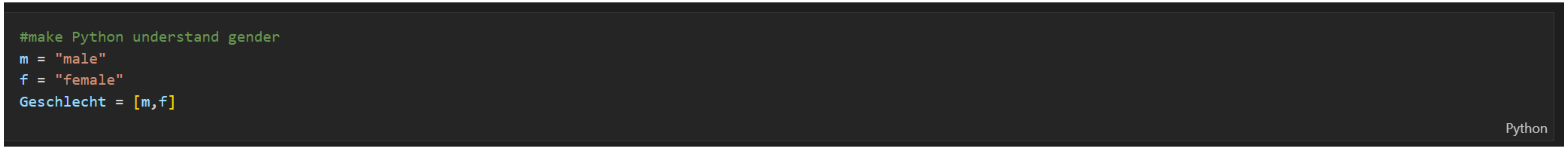
- Geschlechterunterschiede bei 1. August-Reden:
  - Anzahl Ansprachen pro Jahr nach Geschlecht
  - Vokabular: welche Worte werden von welchem Geschlecht wie oft verwendet?
  - Wortgebrauch zwischen den Geschlechtern: Gemeinsamkeiten und Unterschiede

Motivation

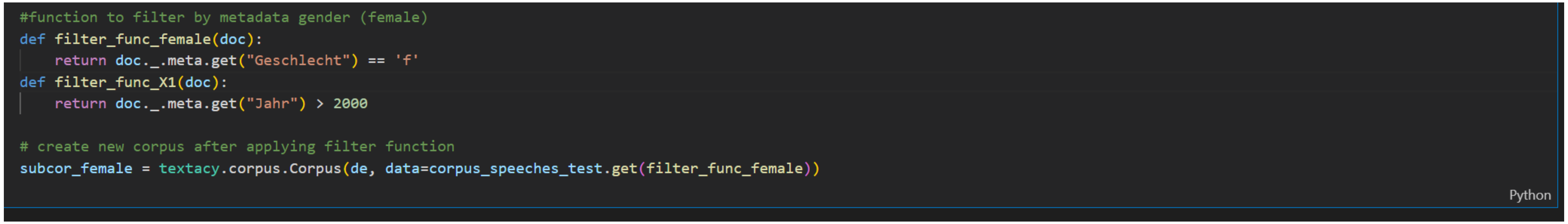
- Aktualität
- Historische Entwicklung
- Rhetorik

Anzahl Ansprachen pro Jahr nach Geschlecht

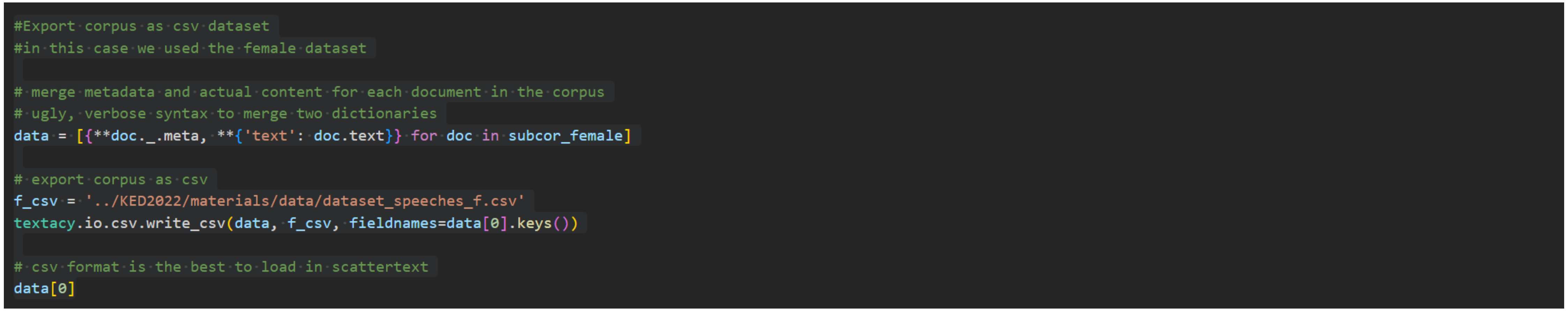
Code am Beispiel weiblicher Sprecherinnen



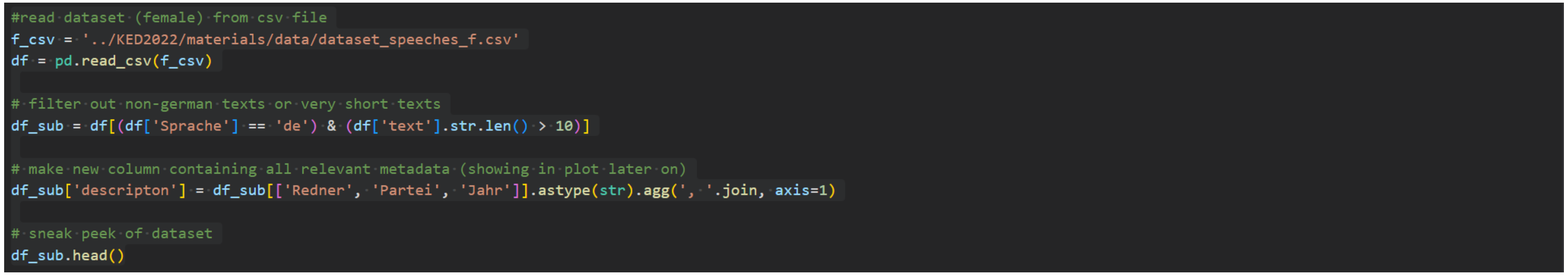
```python
#make Python understand gender
m = "male"
f = "female"
Geschlecht = [m,f]
```



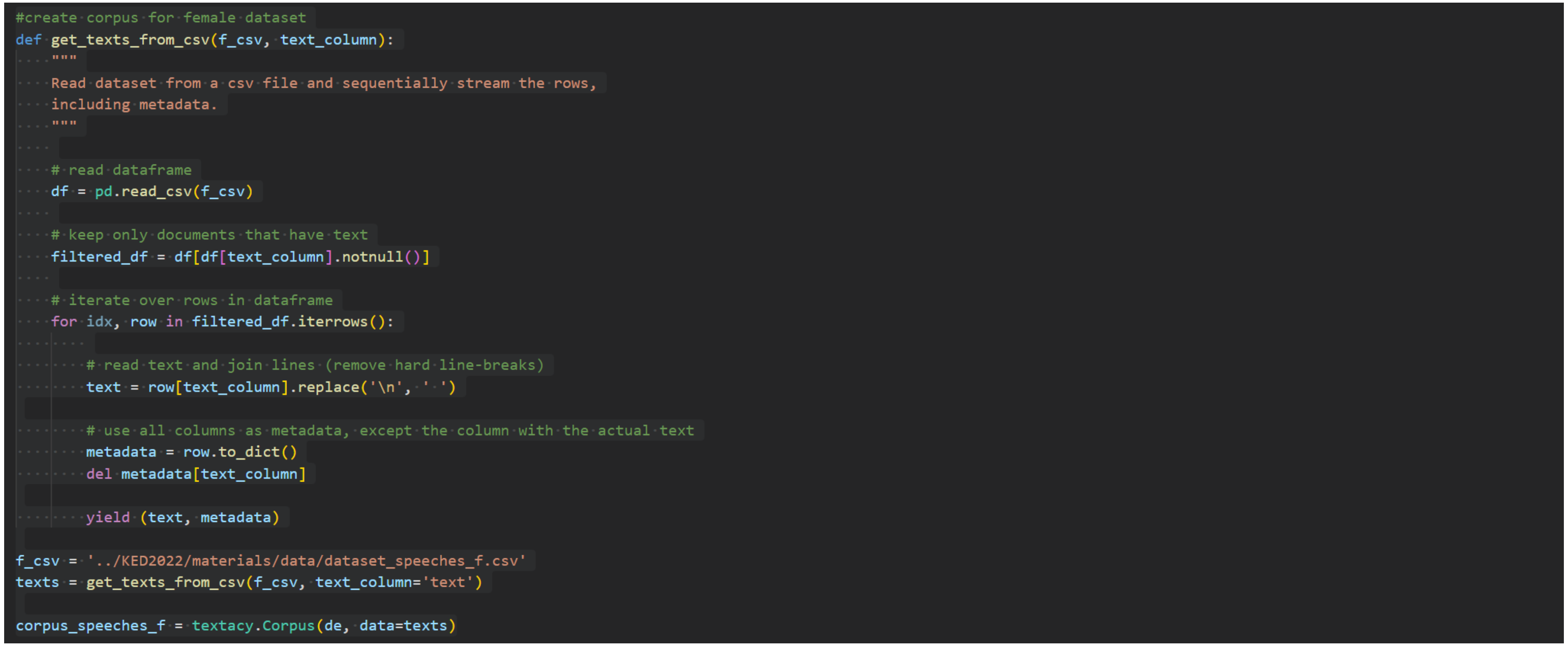
```python
#function to filter by metadata gender (female)
def filter_func_female(doc):
    return doc._.meta.get("Geschlecht") == 'f'
def filter_func_X1(doc):
    return doc._.meta.get("Jahr") > 2000

# create new corpus after applying filter function
subcor_female = textacy.corpus.Corpus(de, data=corpus_speeches_test.get(filter_func_female))
```



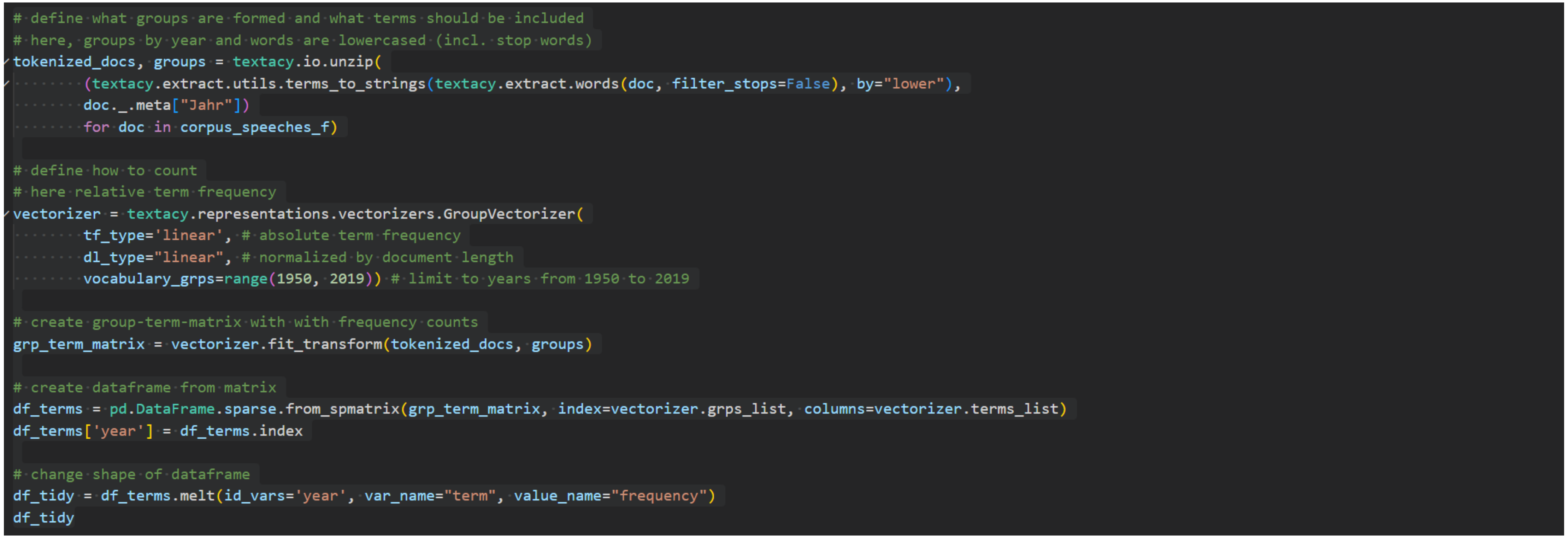
```python
#Export corpus as csv dataset
#in this case we used the female dataset

# merge metadata and actual content for each document in the corpus
# ugly, verbose syntax to merge two dictionaries
data = [{**doc._.meta, **{'text': doc.text}} for doc in subcor_female]

# export corpus as csv
f_csv = '../KED2022/materials/data/dataset_speeches_f.csv'
textacy.io.csv.write_csv(data, f_csv, fieldnames=data[0].keys())

# csv format is the best to load in scattertext
data[0]
```



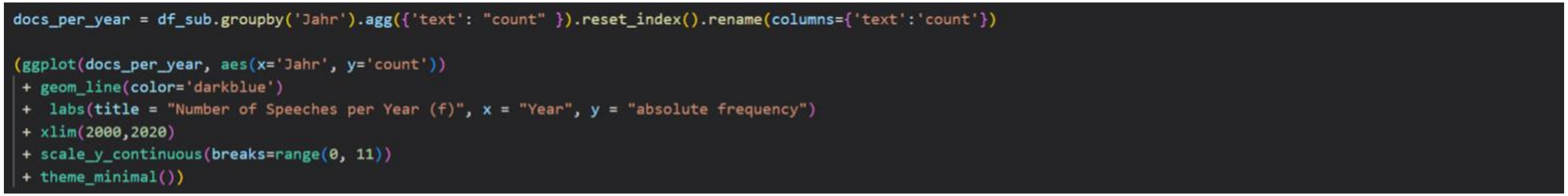
```python
#read dataset (female) from csv file
f_csv = '../KED2022/materials/data/dataset_speeches_f.csv'
df = pd.read_csv(f_csv)

# filter out non-german texts or very short texts
df_sub = df[(df['Sprache'] == 'de') & (df['text'].str.len() > 10)]

# make new column containing all relevant metadata (showing in plot later on)
df_sub['descripton'] = df_sub[['Redner', 'Partei', 'Jahr']].astype(str).agg(', '.join, axis=1)

# sneak peek of dataset
df_sub.head()
```



```python
#create corpus for female dataset
def get_texts_from_csv(f_csv, text_column):
    """
    Read dataset from a csv file and sequentially stream the rows,
    including metadata.
    """
    
    # read dataframe
    df = pd.read_csv(f_csv)
    
    # keep only documents that have text
    filtered_df = df[df[text_column].notnull()]
    
    # iterate over rows in dataframe
    for idx, row in filtered_df.iterrows():
        
        # read text and join lines (remove hard line-breaks)
        text = row[text_column].replace('\n', ' ')

        # use all columns as metadata, except the column with the actual text
        metadata = row.to_dict()
        del metadata[text_column]

        yield (text, metadata)

f_csv = '../KED2022/materials/data/dataset_speeches_f.csv'
texts = get_texts_from_csv(f_csv, text_column='text')

corpus_speeches_f = textacy.Corpus(de, data=texts)
```

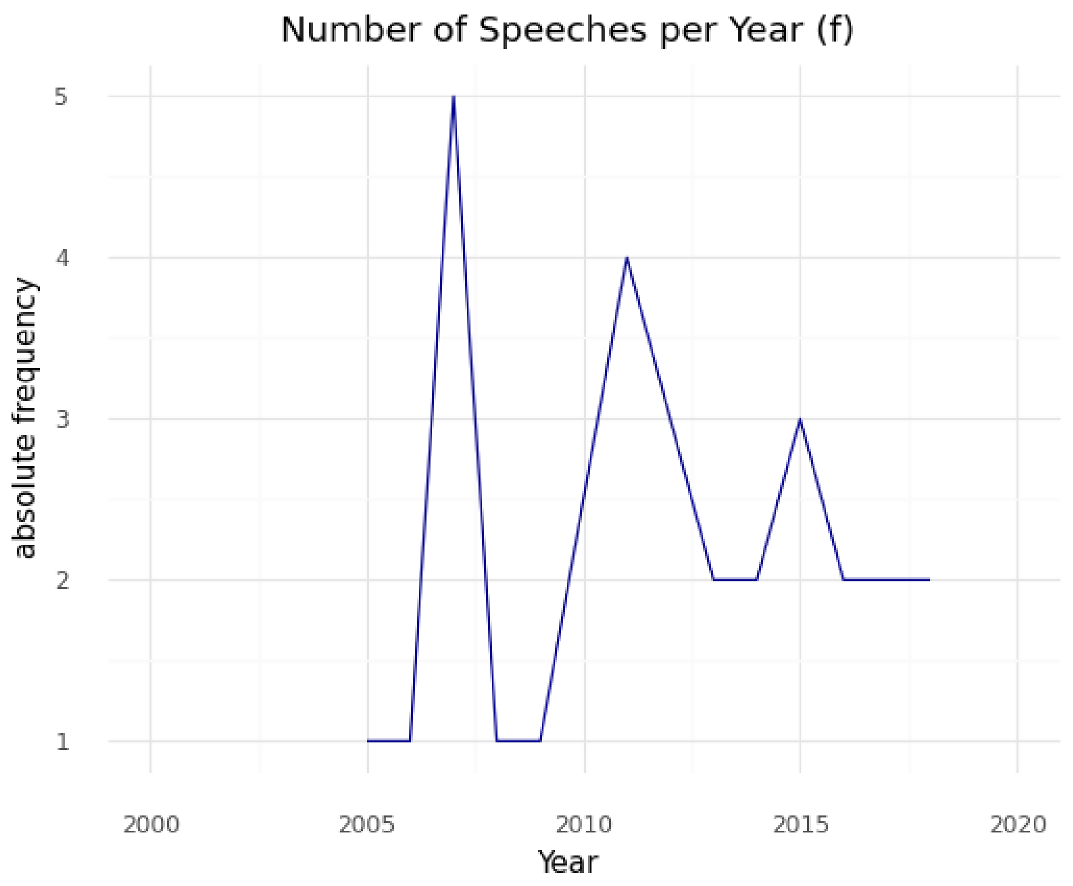


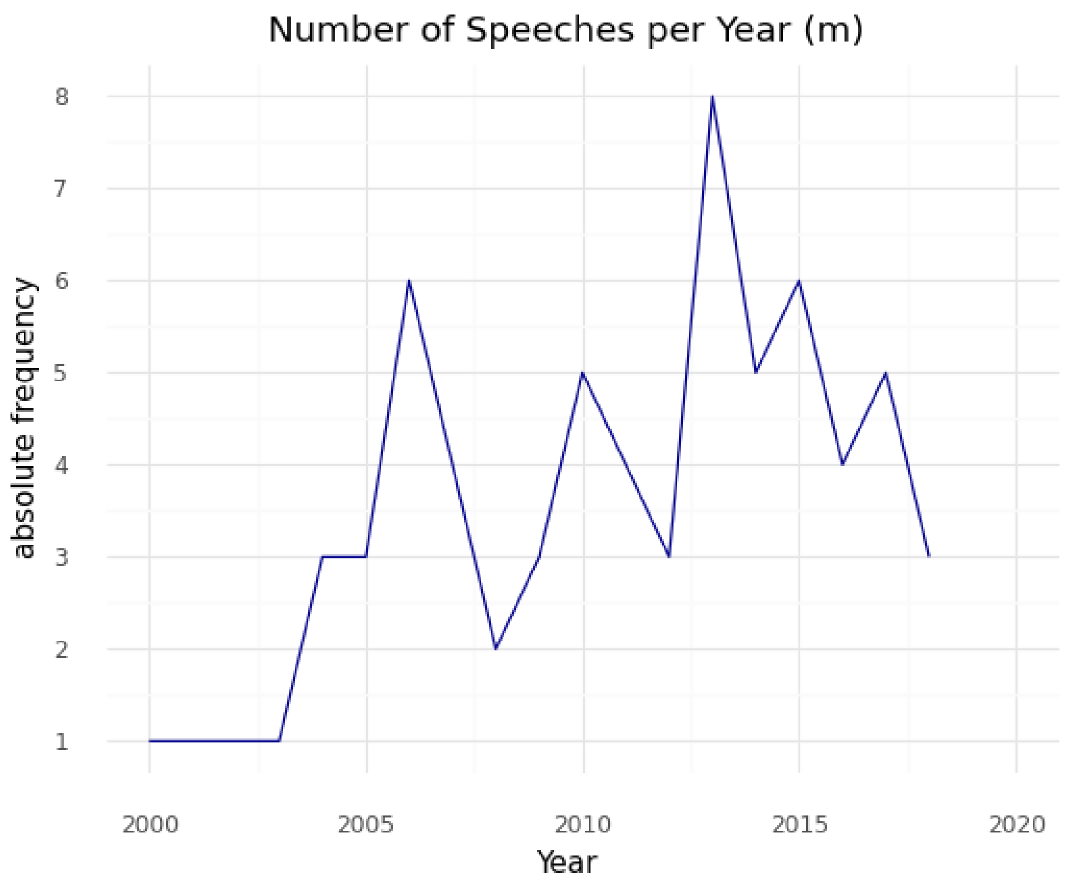
```
# define what groups are formed and what terms should be included
# here, groups by year and words are lowercased (incl. stop words)
tokenized_docs, groups = textacy.io.unzip(
        (textacy.extract.utils.terms_to_strings(textacy.extract.words(doc, filter_stops=False), by="lower"),
        doc._.meta["Jahr"])
        for doc in corpus_speeches_f)

# define how to count
# here relative term frequency
vectorizer = textacy.representations.vectorizers.GroupVectorizer(
        tf_type='linear', # absolute term frequency
        dl_type="linear", # normalized by document length
        vocabulary_grps=range(1950, 2019)) # limit to years from 1950 to 2019

# create group-term-matrix with with frequency counts
grp_term_matrix = vectorizer.fit_transform(tokenized_docs, groups)

# create dataframe from matrix
df_terms = pd.DataFrame.sparse.from_spmatrix(grp_term_matrix, index=vectorizer.grps_list, columns=vectorizer.terms_list)
df_terms['year'] = df_terms.index

# change shape of dataframe
df_tidy = df_terms.melt(id_vars='year', var_name="term", value_name="frequency")
df_tidy
```



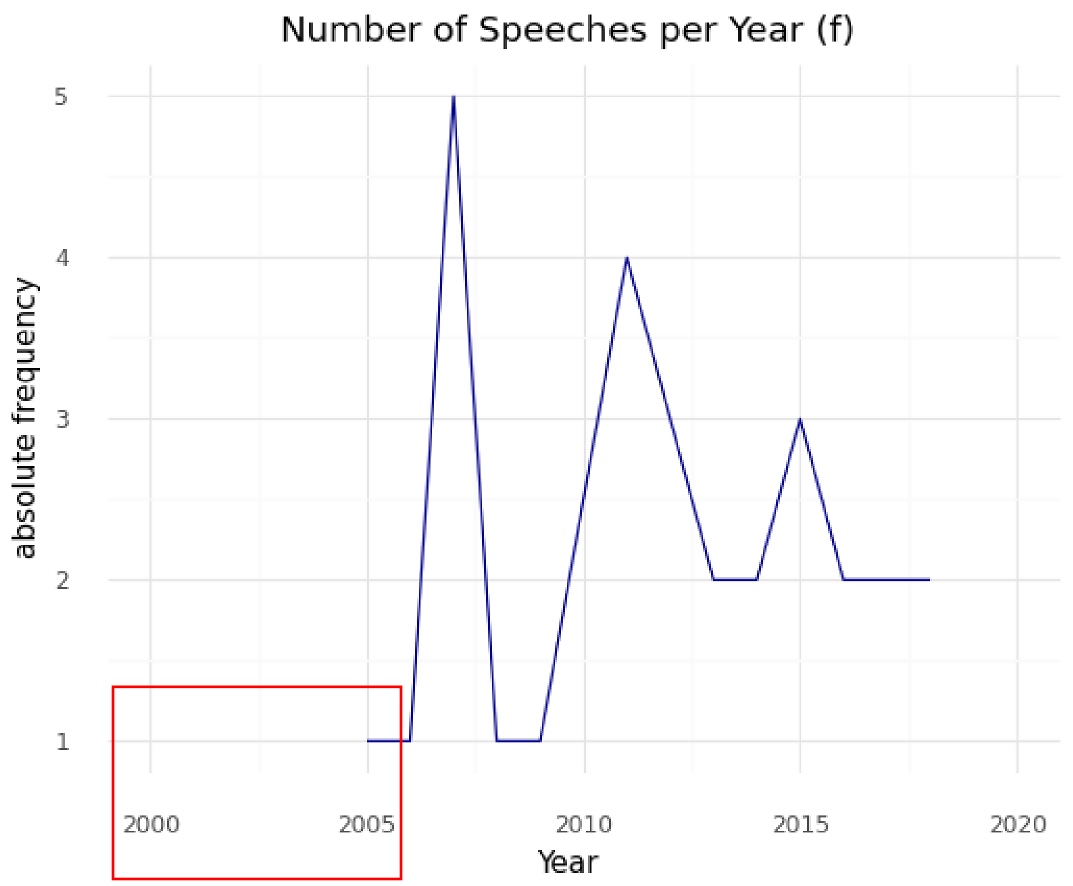
```
docs_per_year = df_sub.groupby('Jahr').agg({'text': "count" }).reset_index().rename(columns={'text':'count'})

(ggplot(docs_per_year, aes(x='Jahr', y='count'))
 + geom_line(color='darkblue')
 +  labs(title = "Number of Speeches per Year (f)", x = "Year", y = "absolute frequency")
 + xlim(2000,2020)
 + scale_y_continuous(breaks=range(0, 11))
 + theme_minimal())
```

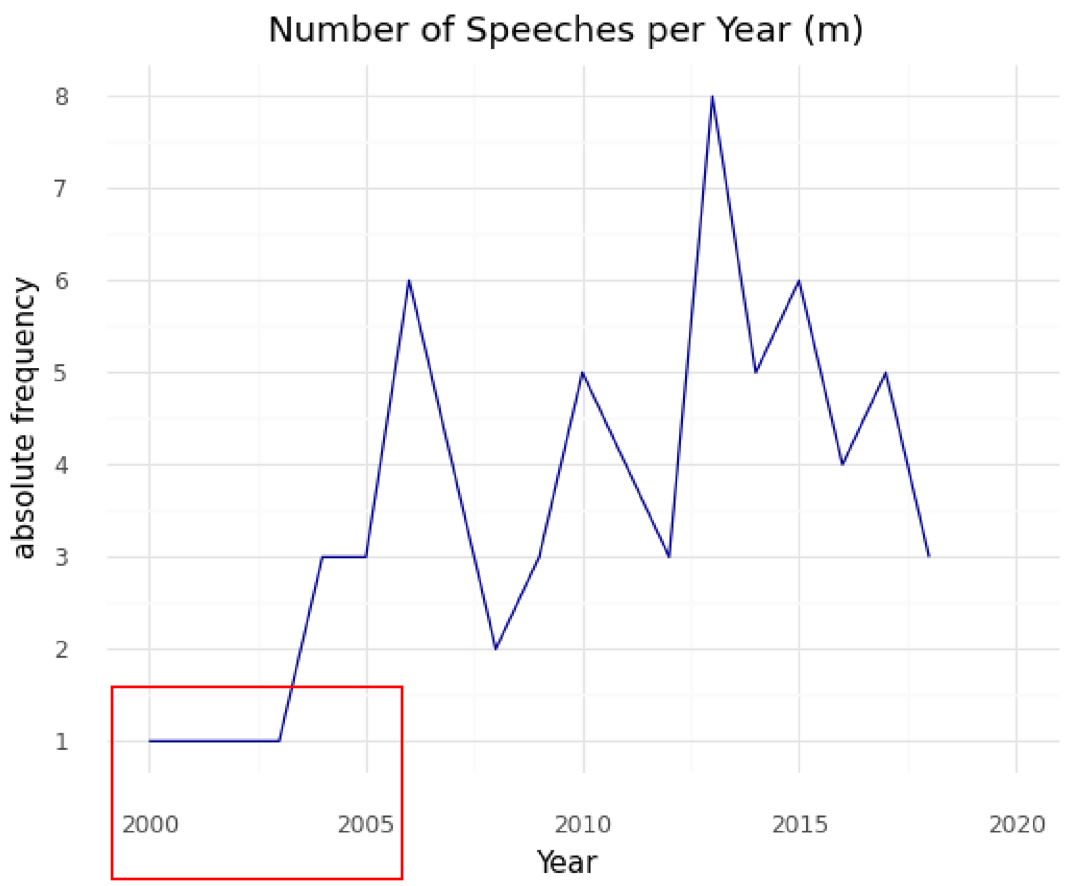
Resultate: Anzahl Ansprachen pro Geschlecht



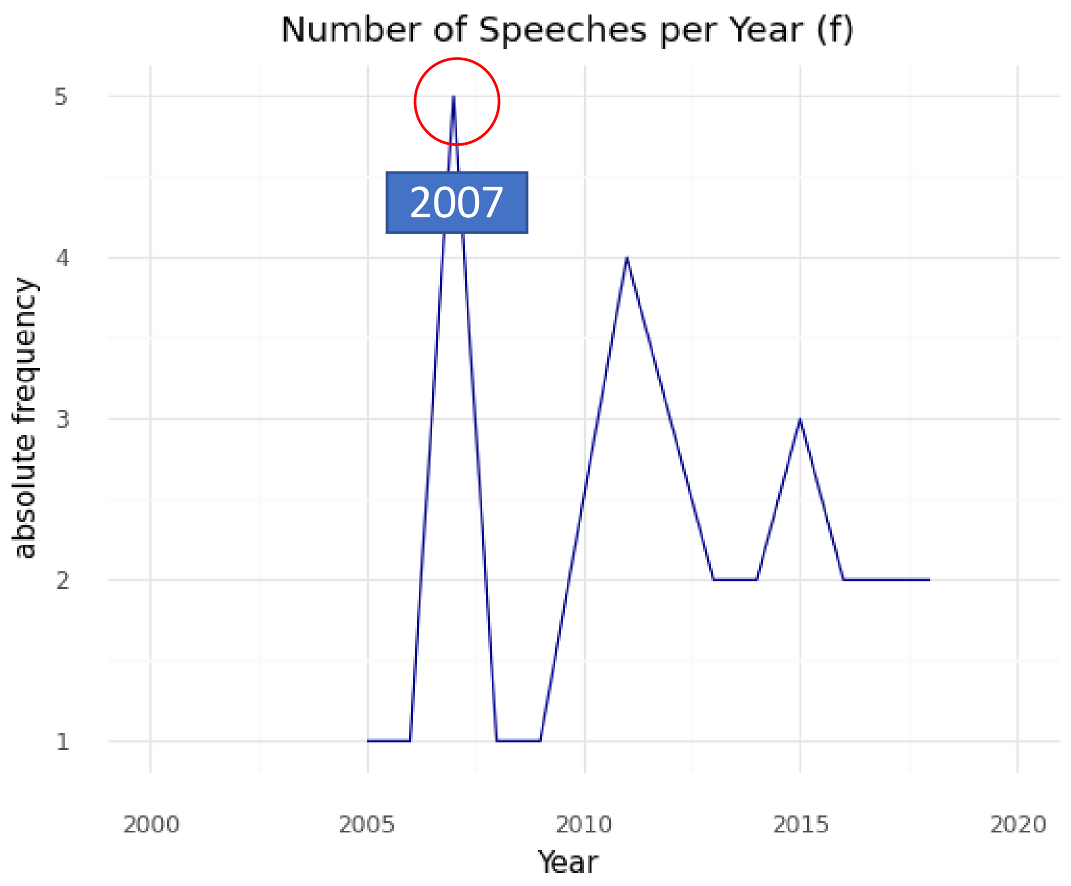


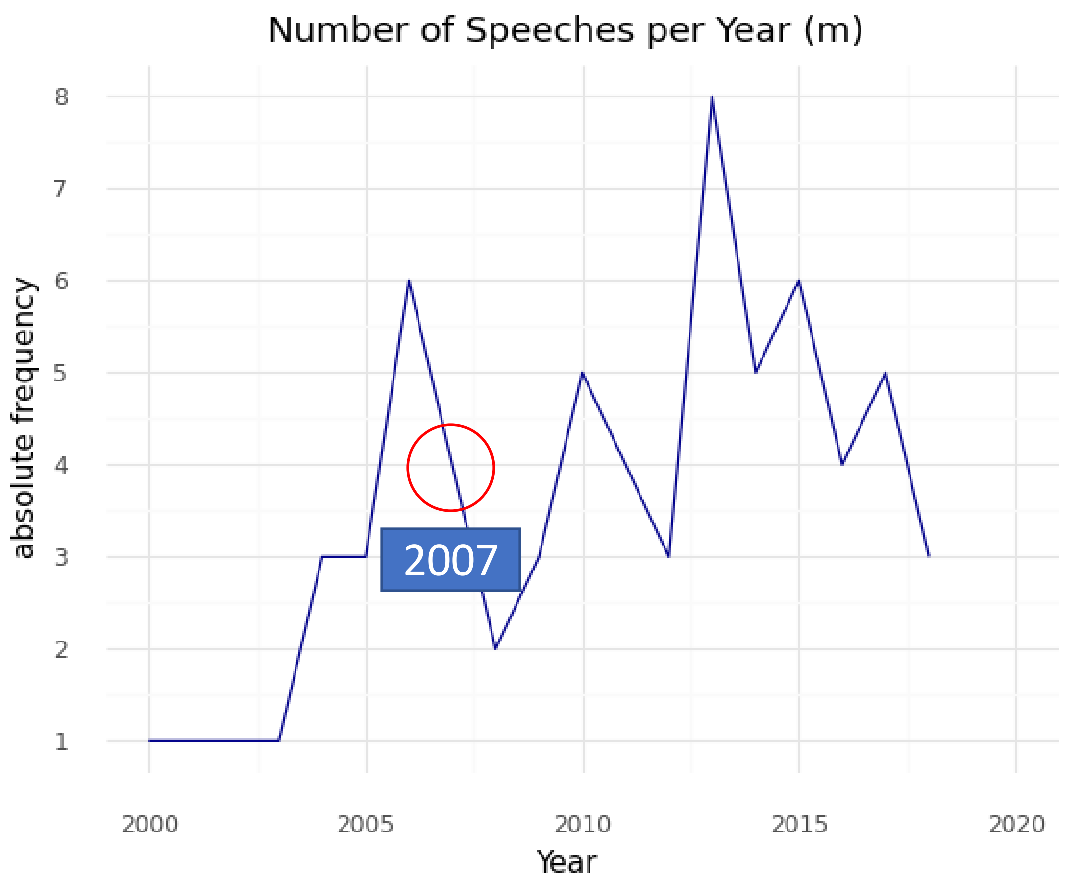
Resultate: Anzahl Ansprachen pro Geschlecht



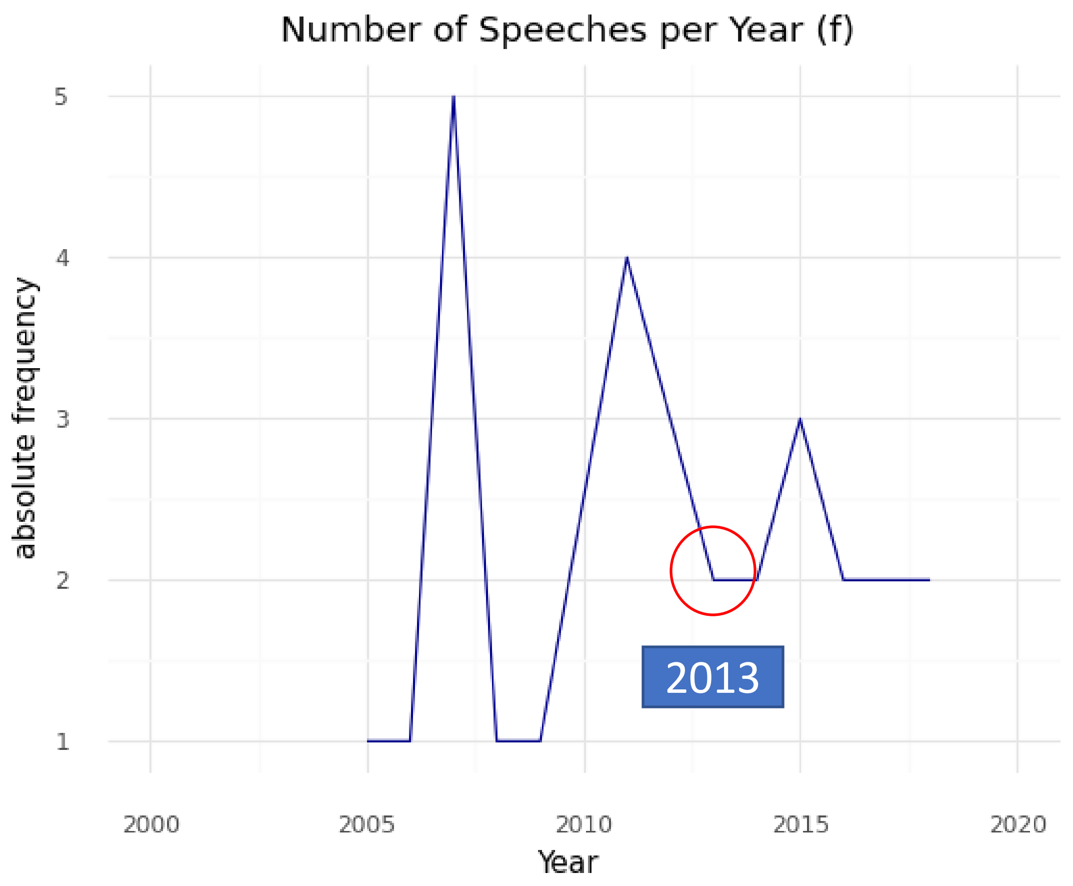


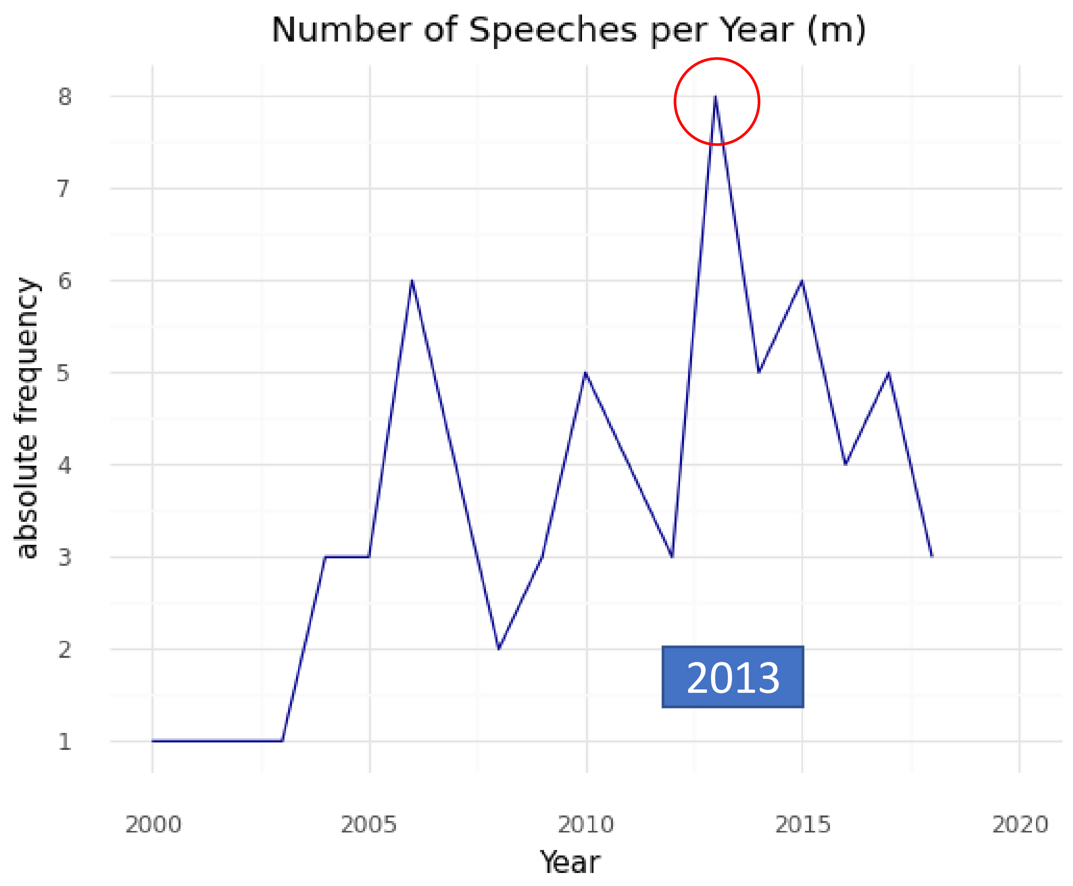
Resultate: Peak Frauen



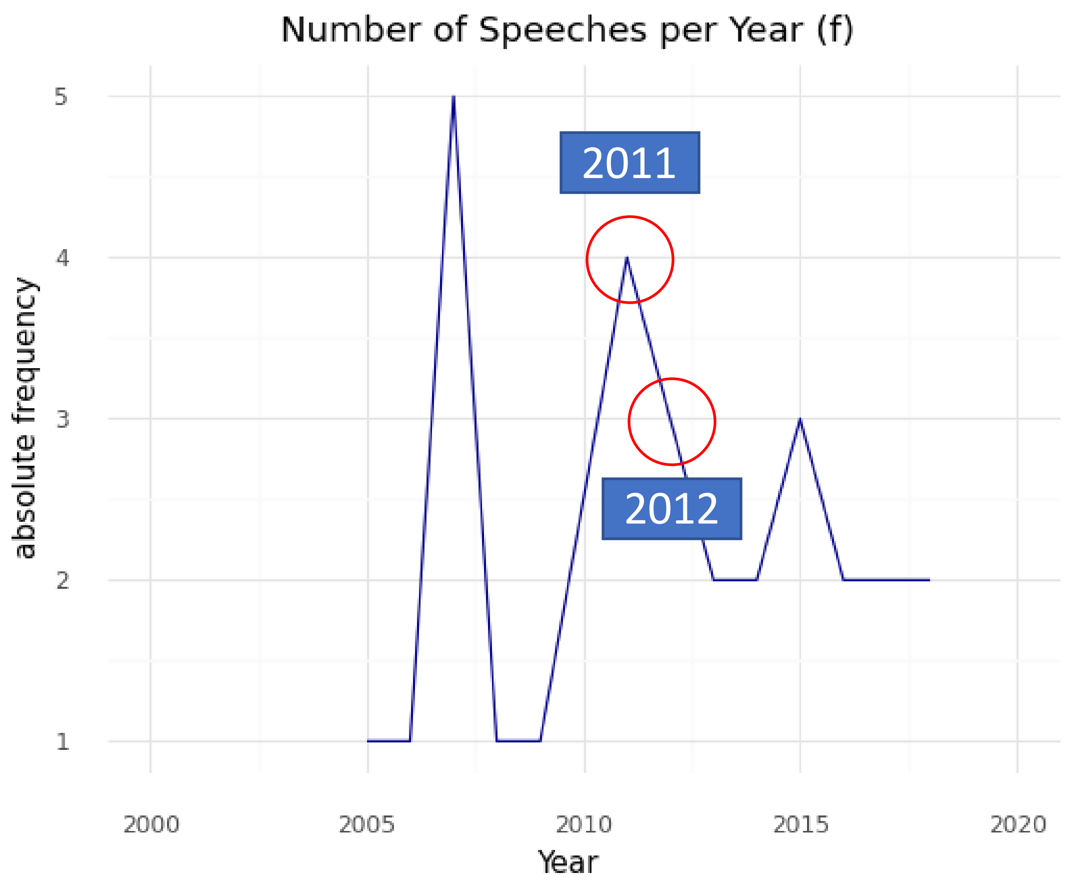


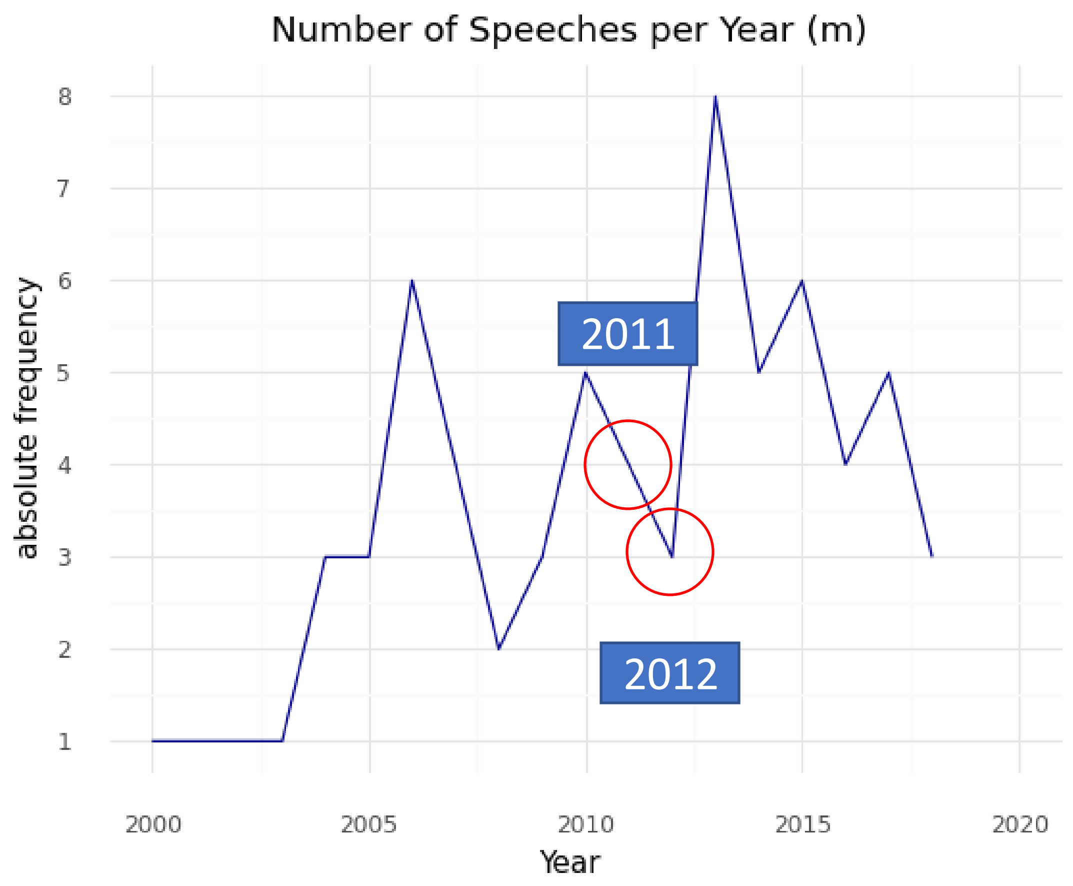
Resultate: Peak Männer





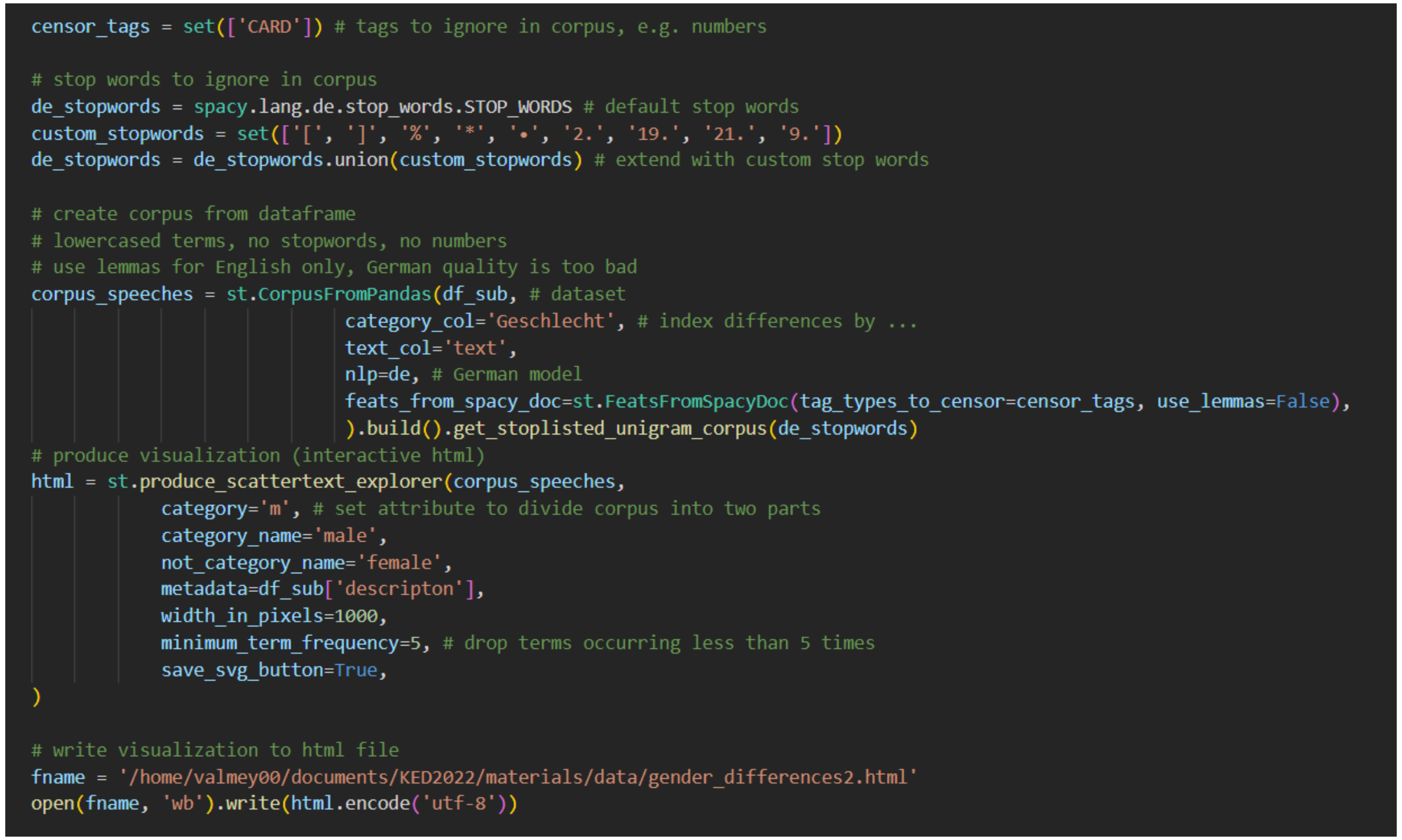
Resultate: Gleichverteilung





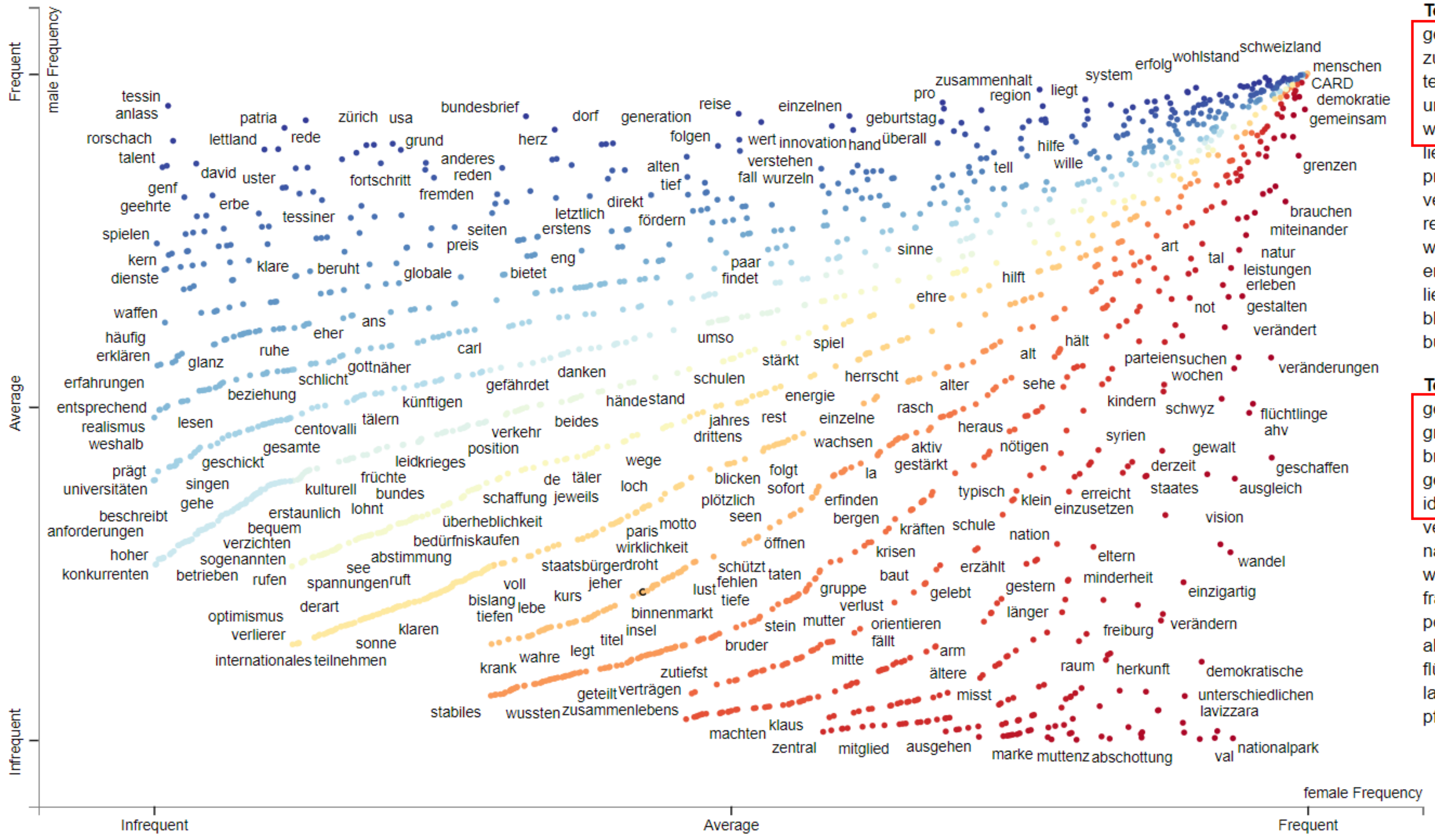
Wortgebrauch nach Geschlecht

Code: Wortgebrauch nach Geschlecht



```
censor_tags = set(['CARD']) # tags to ignore in corpus, e.g. numbers

# stop words to ignore in corpus
de_stopwords = spacy.lang.de.stop_words.STOP_WORDS # default stop words
custom_stopwords = set(['[', ']', '%', '*', '•', '2.', '19.', '21.', '9.'])
de_stopwords = de_stopwords.union(custom_stopwords) # extend with custom stop words

# create corpus from dataframe
# lowercased terms, no stopwords, no numbers
# use lemmas for English only, German quality is too bad
corpus_speeches = st.CorpusFromPandas(df_sub, # dataset
                             category_col='Geschlecht', # index differences by ...
                             text_col='text',
                             nlp=de, # German model
                             feats_from_spacy_doc=st.FeatsFromSpacyDoc(tag_types_to_censor=censor_tags, use_lemmas=False),
                             ).build().get_stoplisted_unigram_corpus(de_stopwords)
# produce visualization (interactive html)
html = st.produce_scattertext_explorer(corpus_speeches,
            category='m', # set attribute to divide corpus into two parts
            category_name='male',
            not_category_name='female',
            metadata=df_sub['descripton'],
            width_in_pixels=1000,
            minimum_term_frequency=5, # drop terms occurring less than 5 times
            save_svg_button=True,
)

# write visualization to html file
fname = '/home/valmey00/documents/KED2022/materials/data/gender_differences2.html'
open(fname, 'wb').write(html.encode('utf-8'))
```

Resultat: Wortgebrauch nach Geschlecht



Top male
- geschichte
- zusammenhalt
- tessin
- unabhängigkeit
- werte
- liebe
- pro
- verantwortung
- rede
- wohlstand
- erfolg
- liegt
- bleiben
- bundesbrief

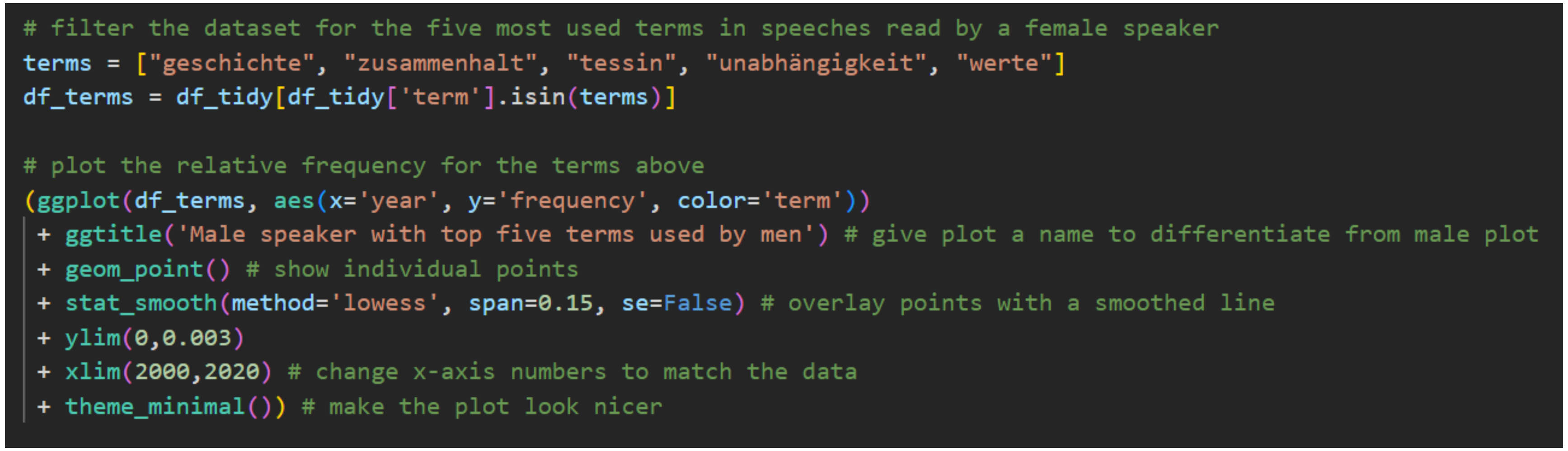
Top female
- gemeinsam
- grenzen
- brauchen
- geschaffen
- identität
- veränderungen
- nationalpark
- wichtig
- frauen
- politische
- ahv
- flüchtlinge
- lavizzara
- pflichten

Characteristic
- CARD
- unseres
- wohlstand
- freiheit
- demokratie
- gemeinsam
- nationalfeiertag
- verantwortung
- unserem
- werte
- unseren
- zusammenhalt
- bundesrat
- zukunft
- vertrauen
- vielfalt
- politischen
- nachbarn
- schweiz
- schliesslich
- generationen
- eidgenossenschaft
- feiern
- offenheit
- braucht
- bewusst
- schweizerinnen
- herausforderungen
- leisten
- minderheiten

Korpus-Umfang: 96 Dokumente (67m, 29f)

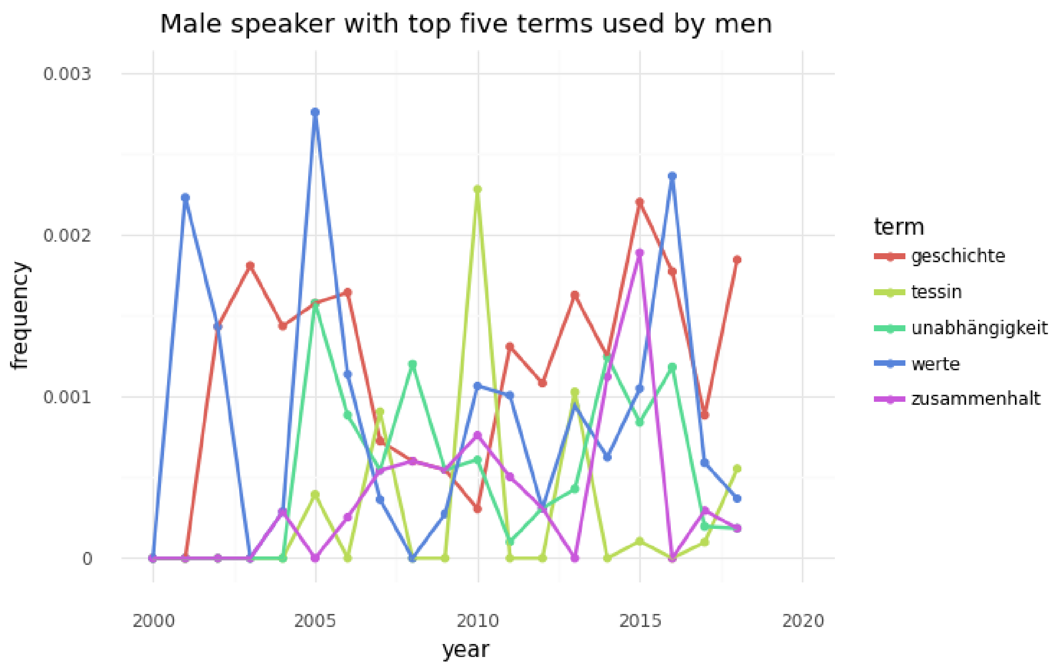
Wortgebrauch zwischen den Geschlechtern

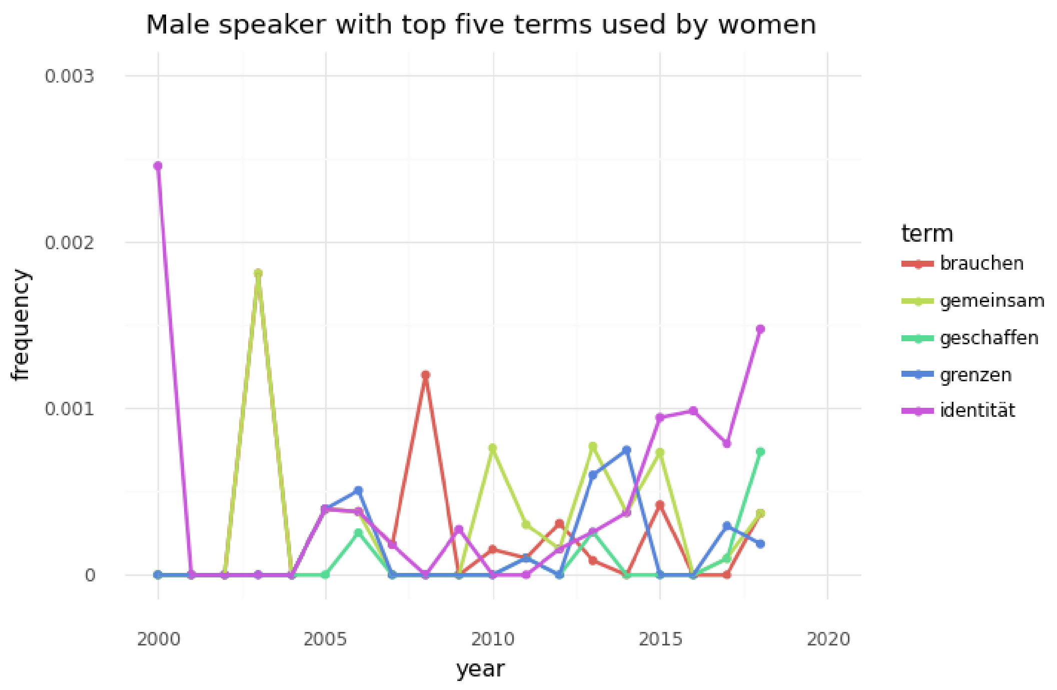
Code: Männer



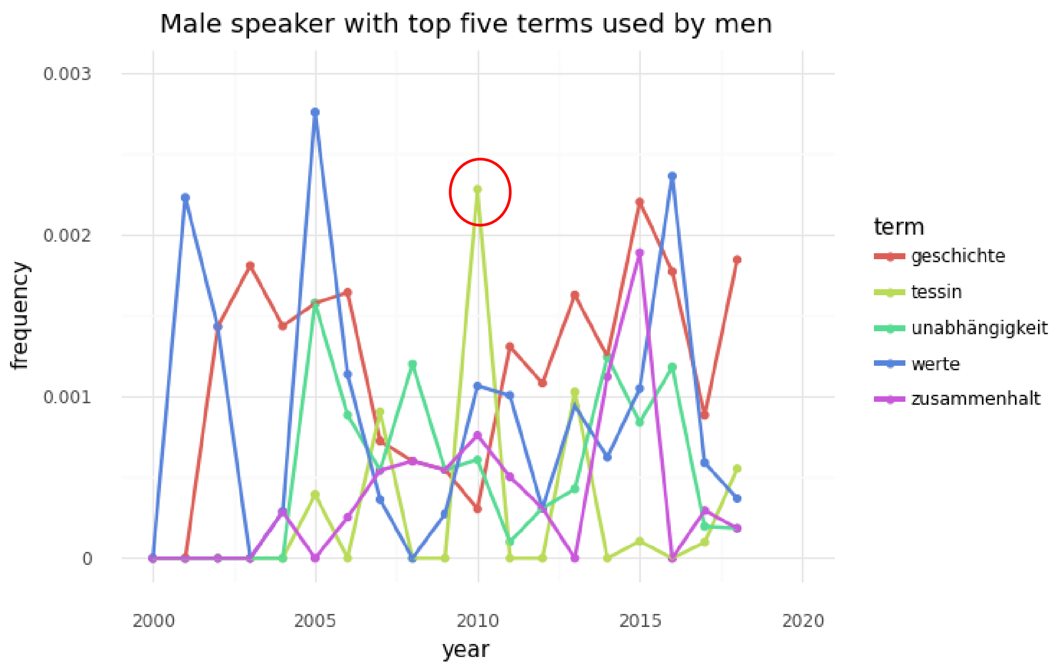
```
# filter the dataset for the five most used terms in speeches read by a female speaker
terms = ["geschichte", "zusammenhalt", "tessin", "unabhängigkeit", "werte"]
df_terms = df_tidy[df_tidy['term'].isin(terms)]

# plot the relative frequency for the terms above
(ggplot(df_terms, aes(x='year', y='frequency', color='term'))
 + ggtitle('Male speaker with top five terms used by men') # give plot a name to differentiate from male plot
 + geom_point() # show individual points
 + stat_smooth(method='lowess', span=0.15, se=False) # overlay points with a smoothed line
 + ylim(0,0.003)
 + xlim(2000,2020) # change x-axis numbers to match the data
 + theme_minimal()) # make the plot look nicer
```

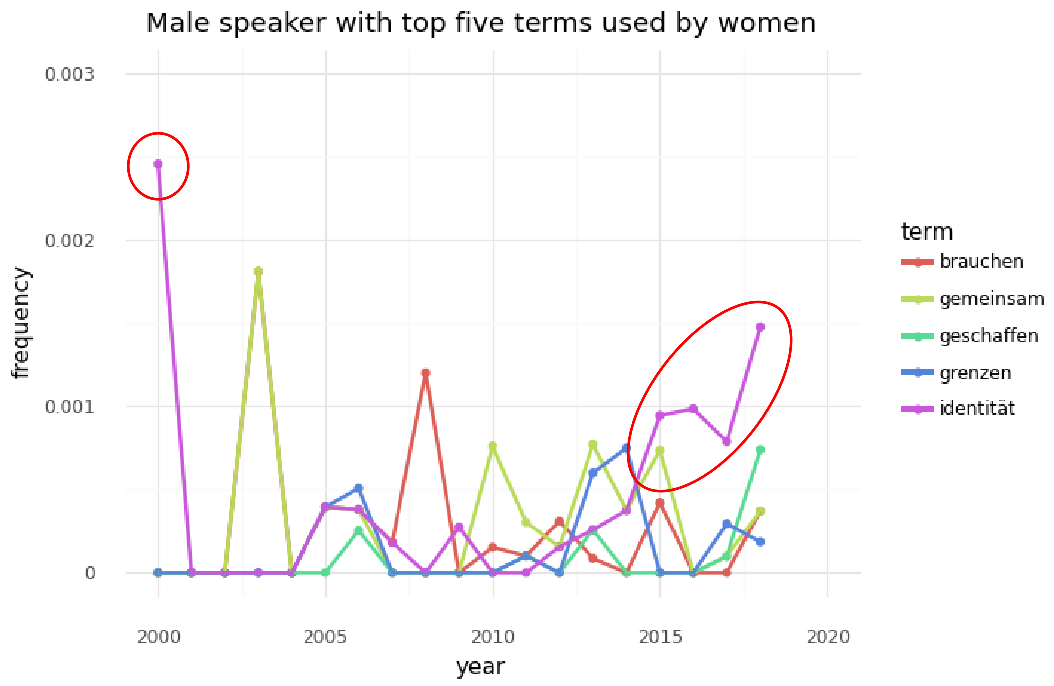
Resultate: Männer



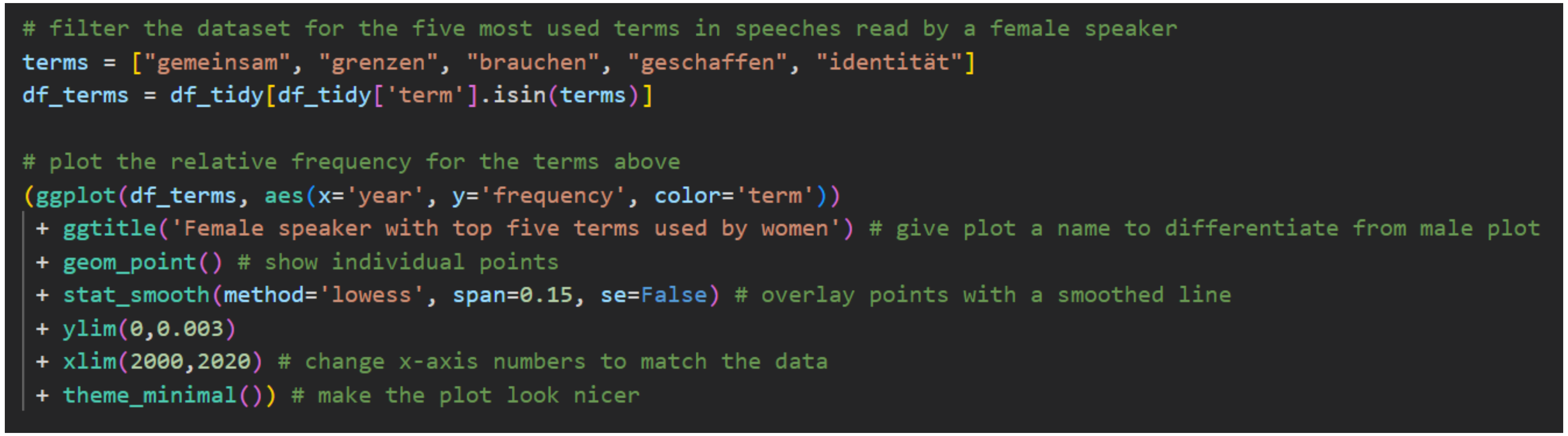


Resultate: Männer



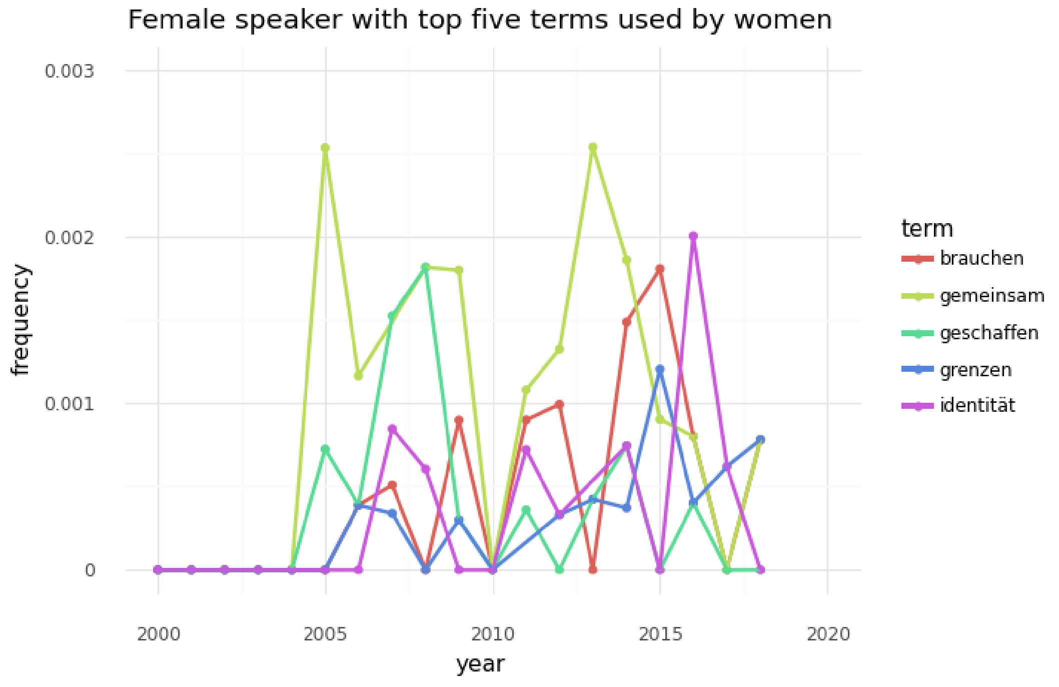


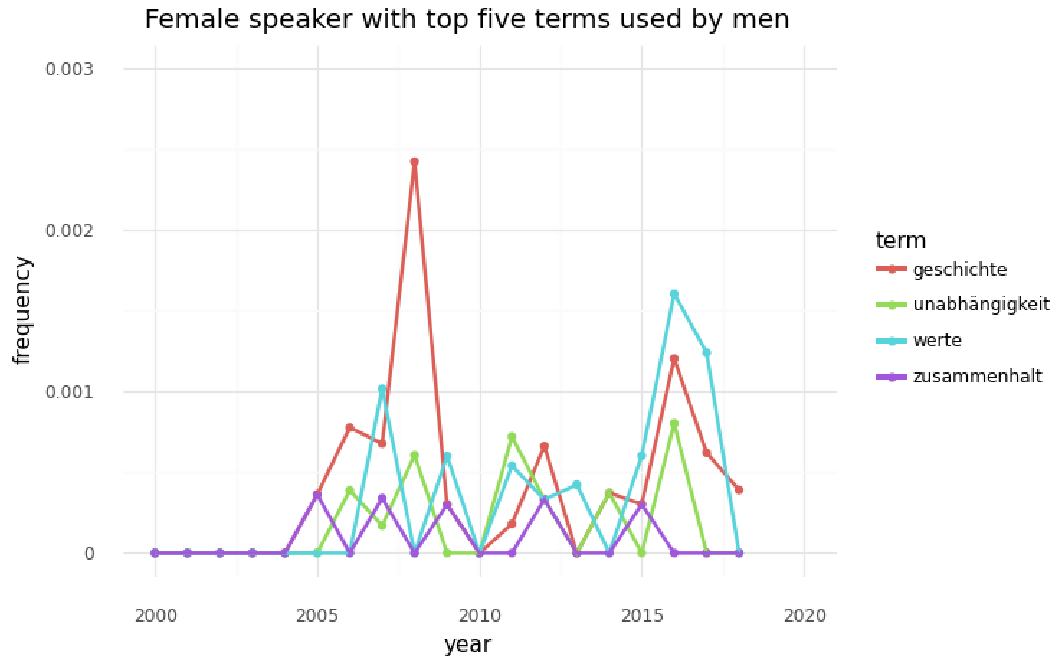
Code: Frauen



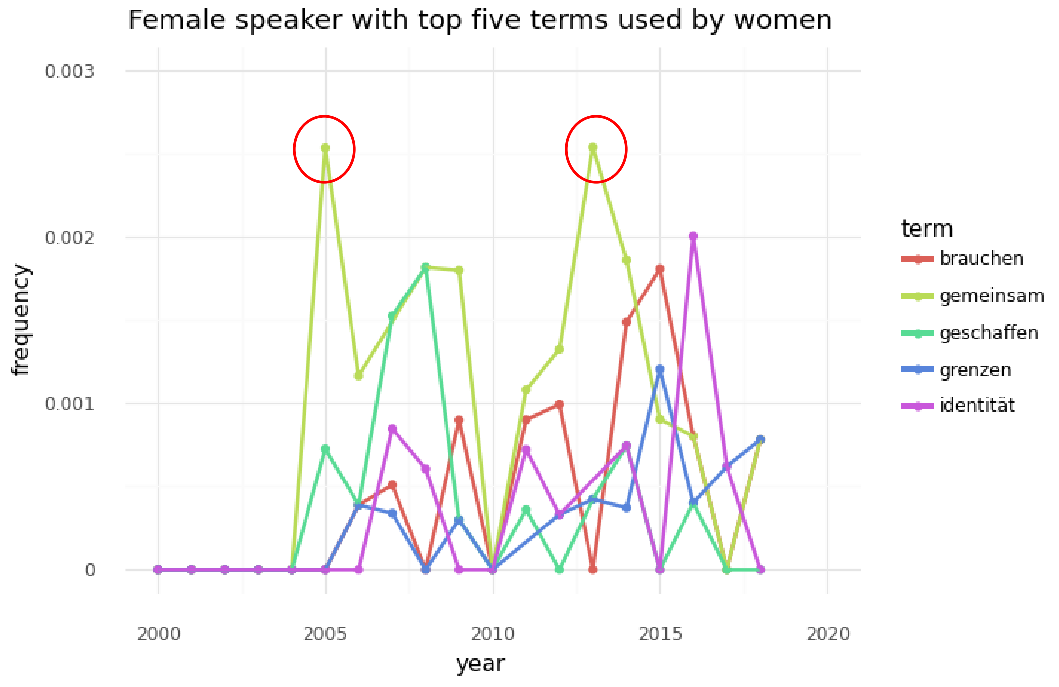
```
# filter the dataset for the five most used terms in speeches read by a female speaker
terms = ["gemeinsam", "grenzen", "brauchen", "geschaffen", "identität"]
df_terms = df_tidy[df_tidy['term'].isin(terms)]

# plot the relative frequency for the terms above
(ggplot(df_terms, aes(x='year', y='frequency', color='term'))
 + ggtitle('Female speaker with top five terms used by women') # give plot a name to differentiate from male plot
 + geom_point() # show individual points
 + stat_smooth(method='lowess', span=0.15, se=False) # overlay points with a smoothed line
 + ylim(0,0.003)
 + xlim(2000,2020) # change x-axis numbers to match the data
 + theme_minimal()) # make the plot look nicer
```

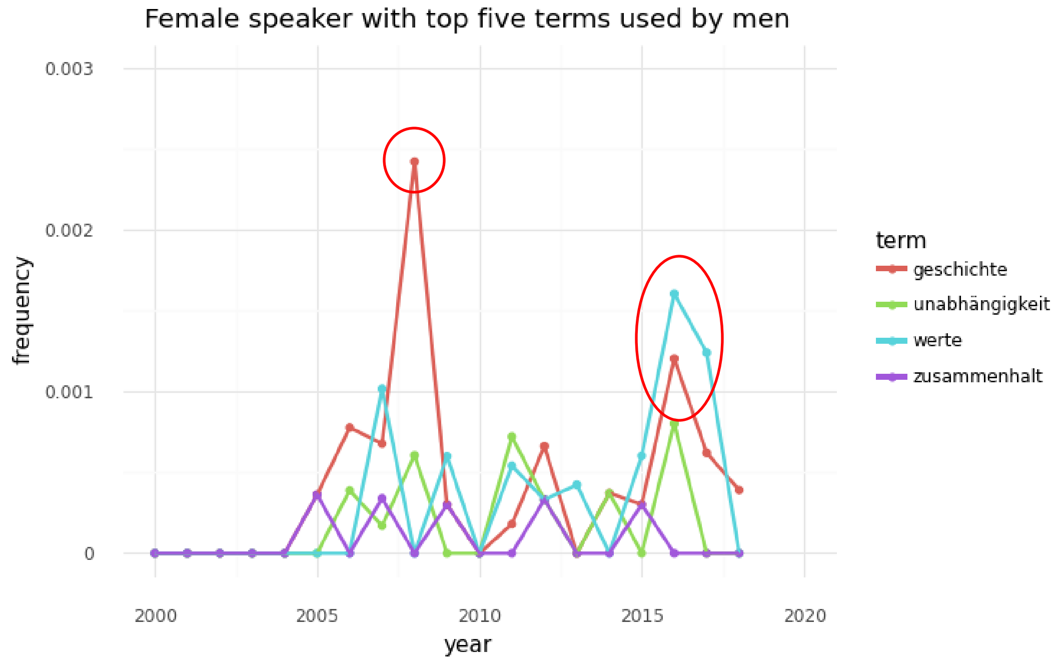
Resultate: Frauen





Resultate: Frauen





Fragen?

Danke für eure Aufmerksamkeit!