flempar : an R-package for analyzing data from the Flemish Parliament

flempar

An R-package for analyzing data from the Flemish Parliament

Evelien Willems*

Frederik Heylen**

September 2023

*University of Antwerp

evelien.willems@uantwerpen.be

**Datamarinier

frederik.heylen@datamarinier.be

Abstract. This note introduces *flempar*, an R-package designed to streamline data acquisition from the Flemish Parliament's open data API. *flempar* provides an interface that allows for easy extraction and linking of data, ranging from details of parliamentary proceedings to legislators' bibliographies. The package aims to reduce the complexity of navigating the numerous API endpoints and the unnesting of the data objects, allowing users to focus on actually working with the data. Implemented in R, the package offers broad applicability for scholars, journalists, and data scientists interested in Flemish parliamentary activities. We provide a comprehensive guide to the package's installation, scope, and data extraction workflow.

Keywords. Belgium, Flemish parliament, legislative studies, parliamentary affairs, R

Acknowledgements. The development of this R-package benefited tremendously from the contributions of Prof. Dr. Wouter Van Dooren (University of Antwerp); the funding of the Department of Political Science, University of Antwerp; and Dr. Evelien Willems wishes to acknowledge the postdoctoral funding by the Research Foundation-Flanders (grantee number: 12A8822N).

Introduction

In the past decade, an extensive range of parliamentary data has become accessible to the general public across multiple countries. Whether accessed via front-end websites or backend Application Programming Interfaces (APIs), researchers focused on parliamentary affairs now enjoy unparalleled ease in obtaining substantial data (examples include the European Parliament, the Norwegian parliament Stortinget, and the UK House of Commons). However, the data harvested from these sources, available in formats such as HTML, XML, and JSON, are usually not designed with statistical analyses in mind. As a result, considerable effort in organization and preprocessing is required before any meaningful analysis can commence.

In recent years, the Flemish Parliament, the parliament of the Flemish region in Belgium, invested in considerable digitizing efforts and made its records of spoken word, documents, and metadata related to all parliamentary activities available via webservices (specifically APIs). Here, the API of the Flemish Parliament is no exception; additional processing is necessary when utilizing their open data for statistical analysis.

Therefore, in this note, we introduce the *flempar* R-package. *flempar* offers a streamlined approach to acquiring data from the Flemish Parliament by tapping into their readily accessible backend API. The data extracted via this package typically require minimal further structuring. The package's purview, as elaborated below, spans from overarching parliamentary information (session details, committees, etc.) to specifics about political parties, legislators' bibliographies, a host of parliamentary activities such as questions and interpellations, bills, hearings, and various other aspects.

With the development of the *flempar* package, we joined several researchers who made a variety of parliamentary data for different countries freely available. Examples include the ParlSpeech V2 data set developed by Rauh and Schwalbach (2020) or the Comparative Legislators Database created by Göbel and Munzert (2022). These examples are, however, different from *flempar* in that they are finished datasets ready for download. The main goal of *flempar* is to allow researchers to easily access any data from the Flemish parliament while offering the flexibility to tailor the data structure to individual research requirements. Most importantly, the package facilitates weaving together different parts (resulting from different endpoints) of the open data API. In this, we follow the approaches by Zumbach and Gföhler (2022), who implemented *'swissparl'*[1], and (Søyland, 2023), who implemented *'stortingscrape'*[2] for the Norwegian parliament.

[1] See https://CRAN.R-project.org/package=swissparl
[2] See https://CRAN.R-project.org/package=stortingscrape

In this note, we first delve into the use of the open data of the Flemish parliament. From there, we navigate into the details of the *flempar* package, encompassing its scope and application while delivering a concise snapshot of its installation process and the workflow for extracting data.

Open Data and the Flemish Parliament

Open data initiatives by government bodies could significantly increase transparency, accountability, citizen engagement, and social innovation (Aikins & Krane, 2010). However, research shows that practical and technical implementation may pose barriers to the uptake of open data (Cahlikova & Mabillard, 2020).

Since 2015, the Flemish Parliament has offered data from its parliamentary database through an API, which is freely and openly accessible to both citizens and businesses. The data include various kinds of information related to parliamentary activities and are available without restrictions, although users must attribute the Flemish Parliament as the data source when reusing the information. Importantly, no copyright restrictions exist on official parliamentary documents or speeches made and their associated metadata (see *Gratis Open Data Licentie Vlaanderen)*. Note that some restrictions on footage and graphical elements exist. Furthermore, under the obligation, each user should read the license[3].

flempar was built and tested under version 2.7.29 of the open data API[4]. The API can be subjected to change. It mirrors all the data in the interface. The API can be easily tested via swagger. However, the documentation is limited to this swagger. The Flemish Parliament also exposes a *search* API[5]. The API returns JSON or XLM. Below, we present an example in JSON:


```
{
  "disclaimer":" ",
  "items": [
    {
      "volksvertegenwoordiger": {
        "deelstaatsenator": false,
        "fotowebpath": "https://docs.vlaamsparlement.be/pfile?id=1695546",
        "fractie": {
          "id": 1279907,
          "kleur": "83de62",
          "logo": "https://docs.vlaamsparlement.be/pfile?id=1866773",
          "naam": "Groen",
          "zetel-aantal": 14
        },
        "id": 4544,
```


[3] See https://www.vlaamsparlement.be/nl/parlementair-werk/dossiers/dossiers/open-data
[4] See http://ws.vlpar.be/e/opendata/api/
[5] See http://ws.vlpar.be/api/swagger/#!/search/getSearchResult

```
        "is-huidige-vv": "J",
        "kieskring": "Kieskring Antwerpen",
        "link": [
          {
            "href": "http://ws.vlpar.be/e/opendata/vv/4544?lang=nl",
            "rel": "self",
            "templated": false,
            "variableNames": [],
            "variables": []
          }
        ],
        "naam": "Aerts",
        "voornaam": "Staf",
        "zetel": "127"
      }
    } ...
```


Implementing an open data policy can be difficult for political institutions. They must balance upholding ideals like transparency and accountability while safeguarding against the risk of releasing certain types of data (e.g., minutes of closed-door meetings and confidential information). Furthermore, budgetary constraints may weigh on how elaborate the open data solution can be and what technical background is required to use it. Moreover, the sheer breadth of data spanning multiple decades and various policy areas presents challenges and often results in a flurry of endpoints and complex data structures characterized by deeply nested information (i.e., lists in lists). All this means social science researchers might face a high threshold for using open data.

Philosophy, scope, and usage of *flempar*

The *flempar* package is designed to serve as a robust and streamlined interface between users and the API of the Flemish Parliament, encapsulating a commitment to both open data and academic rigor. To do so, the complexities of navigating the myriad of different API endpoints and parsing JSON/XML objects are mitigated. Built for R, one of the most popular programming languages for statistical computing, *flempar* integrates seamlessly into the R ecosystem.

One of our primary objectives is to maximize the ease with which the API's output can be integrated into existing R libraries, particularly by translating it into tabular data that most R libraries commonly expect. The idea is to allow users to focus on what is most important: analyzing the data. Moreover, with *flempar* we aim to enhance the reproducibility of research, a foundational element of scientific credibility. Users can repeat analyses without sharing files with just a few lines of code.

In each data frame *flempar* returns, the rows correspond to a sensible unit of analysis. For instance, when retrieving the current members of parliament (MP), each row represents one MP. Hence, instead of a data frame where each row corresponds to all the combinations of the nested values (like a row for each MP, for each job they had and each degree obtained, all commissions they have seat in, etc.), the package provides – as much as possible - a flat, unidimensional data frame. To do so, we added parameters to the functions to get specific results (see below). Though, in some instances, you will find additional variables, lists or data frames nested within the cells.

We integrate this package to use with the popular *tidyverse* libraries. Specifically, as data type, we return tibbles as data frames. Also, *flempar* functions can be used inside *dplyr* chains (see below). Furthermore, all data frames contain the necessary native ID (the IDs found within the API), allowing the user to link the datasets. For instance, linking the MP to an excerpt of speech in a debate. Lastly, we invested in normalizing the data, for example, by dealing with inconsistencies when collecting data from different periods. For instance, depending on the timeframe, the API for written questions can return Word, PDF, or PDF with two columns of text or RTF. *flempar* parses all this data.

flempar leverages parallel computing to accelerate data retrieval, particularly for tasks that require multiple underlying API calls to aggregate the final result. Distributing these calls across multiple processing threads dramatically reduces the time needed for data acquisition. This solution is achieved by integrating the FOREACH package, which efficiently manages parallel execution. The speed-up depends on your machine's processing capabilities; with up to ten cores, for instance, you could receive results in as little as one-tenth of the time it would take using a single-threaded approach.

Currently, the package covers many of the API's resources, but not all. Notably, it does not include petitions to parliament (red. *verzoekschriften*). *flempar* covers resources related to members of parliament, written questions, the details of the proceedings in plenary sessions and committees (including meeting details, voting results, spoken word of questions and interpellations, and debates), and comprehensive information regarding the legislative work (parliamentary initiatives). The API contains historical dating back to 1971 (when the parliament was known as the "*Cultuurraad voor de Nederlandse Cultuurgemeenschap"*). However, not all resources extend this far into the past, and non-digital data remains not accessible. Most resources are available, typically stretching back to the early 2000's. The package aims to be comprehensive but will continue to evolve, with planned updates to include even more data. A helpful tip is to use the interface to check the resulting data frames since the Parliament

uses its own API to build their website. The IDs returned in the data frame should match those used in the interface https://www.vlaamsparlement.be/nl/parlementair-werk/.[6]

A note on error logging. When an error occurs (e.g., because the server fails), the execution is stopped. In this way, no erroneous or incomplete results are shown. One notable exception is when using the document function. Here, when retrieval of a document goes wrong, the 'text' field of the document shows the error instead of stopping the execution. Take this into account when parsing this text. Operationally, for large data retrieval jobs, we suggest making a script that divides the load into smaller chunks and uses try-catch. For example, see: https://www.flempar.be/post/longer-time-periods/.

Finally, given the large amount of different data types that can be queried, it is not always evident which data types can be meaningfully retrieved. For instance, debates are only held in plenary and never during committee meetings. Or, written questions are never associated with spoken word (obviously). Here, you can check the "possibilities matrix" to assess which parameter specifications are compatible and which are not.

Installation

You can install the *flempar* package from the GitHub account of the Political Science Department of the University of Antwerp, https://github.com/PolscienceAntwerp/flempar. The first line of code below checks whether you have devtools installed and installs it if not. The second line installs the *flempar* package. The website contains the most frequent installation problems and how to solve them.

```
if (!requireNamespace("devtools", quietly = TRUE)) {
  install.packages("devtools")
}

devtools::install_github("PolscienceAntwerp/flempar")

library(flempar)
```

For certain operations, *flempar* uses your local Word installation. If this is not installed, it uses LibreOffice. *flempar* will still work without Word or LibreOffice but will issue warnings in the rows of data that could not be extracted due to missing software. If you want to use all the functionalities, install Word or LibreOffice.

[6] For instance, a meeting with ID '1356685', can be visualized with https://www.vlaamsparlement.be/nl/parlementair-werk/commissies/commissievergaderingen/1356685.

If you experience issues while installing *flempar*, or if you prefer containerized environments, then you may opt for the containerized version via Docker. After installing Docker on your system, run the following commands in a command prompt. Open a browser and visit localhost:8787 to find a RStudio server with *flempar* installed. This Docker image contains all the necessary dependencies (including LibreOffice).

```
docker pull datamarinier/flempar docker run --rm -ti -e PASSWORD=yourpassword
-p 8787:8787 datamarinier/flempar
```

Data extraction workflow

In this section, we discuss some examples of data extraction with *flempar*. The basic data extraction from the Flemish Parliament's API revolves around various forms of ID tags. For example, all MPs have a unique ID, all parliamentary activities have unique IDs, all votes have unique IDs, and so on. The package contains two core functions to tap into the specifics about political parties, legislators' bibliographies, a host of parliamentary activities such as questions and interpellations, bills, hearings, and various other aspects covering decades of plenary and committee sessions and various policy areas:

- Key characteristics of Flemish members of parliament (MPs): get_mp()
- Parliamentary work, including speeches and documents: get_work()

While the get_mp function departs from individual legislators' attributes to then extend to the activities they are engaged in, the network function revolves around the parliamentary activities themselves. In many instances, data obtained from either function can be seamlessly integrated with data acquired from the other function.

We start by showing one of the most straightforward queries you can perform using the get_mp() function, the extraction of the current Flemish parliamentarians (MPs) and their demographics. Our query results in a data frame with 124 observations and 21 variables, precisely the number of MPs currently in office.

```
library("flempar")
mp_bio <- get_mp(selection="current", fact= "bio")
mp_bio
```

The data structure looks like this.

```
# A tibble: 124 × 21
   id_mp voornaam   achternaam  geslacht geboortedatum geboorteplaats
   <int> <chr>      <chr>       <chr>    <date>        <chr>
 1  4544 Staf       Aerts       M        1981-09-07    Duffel
 2  4381 Meyrem     Almaci      V        1976-02-24    Sint-Niklaas
 3  4382 Els        Ampe        V        1979-01-17    Oostende
 4  4383 Hannes     Anaf        M        1985-03-03    Turnhout
 5  4052 Imade      Annouri     M        1984-02-19    Antwerpen
 6  4384 Roosmarijn Beckers     V        1986-10-22    Sint-Truiden
 7  4385 Stijn      Bex         M        1976-06-14    Diest
 8  4386 Adeline    Blancquaert V        1996-05-23    Gent
 9  3430 Robrecht   Bothuyne    M        1979-11-15    Gent
10  3426 Karin      Brouwers    V        1964-07-26    Leuven
```

Besides querying for the current set of MPs in office, the selection parameter can be changed to ‘former’ or ‘date’. And next to their demographics (fact= “bio”), data about their career, education, presences in commissions and plenary, and party affiliations can be obtained.

Next, we move to the extraction of period-specific data. Note that the legislatures are accessed through a custom function.

```
get_legislatures()
```

```
# A tibble: 14 × 5
      id start               eind                naam      verkiezingsdatum
   <int> <dttm>              <dttm>              <chr>     <chr>
 1     1 1985-12-02 23:00:00 1987-12-11 23:00:00 1985-1988 1985-10-13T00:00:00+0100
 2     2 1988-02-01 23:00:00 1991-11-22 23:00:00 1988-1992 1987-12-13T00:00:00+0100
 3     3 1992-01-06 23:00:00 1995-05-19 22:00:00 1992-1995 1991-11-24T00:00:00+0100
 4     4 1995-06-12 22:00:00 1999-06-11 22:00:00 1995-1999 1995-05-21T00:00:00+0200
 5     5 1999-07-05 22:00:00 2004-06-11 22:00:00 1999-2004 1999-06-13T00:00:00+0200
 6     6 1981-12-21 23:00:00 1985-10-11 23:00:00 1981-1985 1981-11-08T00:00:00+0100
 7     7 1979-01-17 23:00:00 1981-11-06 23:00:00 1979-1981 1978-12-17T00:00:00+0100
 8     8 1977-05-11 22:00:00 1978-12-15 23:00:00 1977-1979 1977-04-17T00:00:00+0200
 9     9 1974-04-03 23:00:00 1977-04-15 22:00:00 1974-1977 1974-03-10T00:00:00+0100
10    10 1971-12-06 23:00:00 1974-03-08 23:00:00 1971-1974 1971-11-07T00:00:00+0100
11    11 2004-07-05 22:00:00 2009-06-05 22:00:00 2004-2009 2004-06-13T00:00:00+0200
12    12 2009-06-29 22:00:00 2014-05-23 22:00:00 2009-2014 2009-06-07T00:00:00+0200
13    13 2014-06-16 22:00:00 2019-05-24 22:00:00 2014-2019 2014-05-25T00:00:00+0200
14    16 2019-06-17 22:00:00 2023-09-23 22:00:00 2019-2024 2019-05-26T00:00:00+0200
```

Indeed, the Flemish parliament’s API stores information going back decades. Using this info, we can, for example, query the demographics of Flemish MPs in office in 1995. Note that we specified the selection parameter to ‘date’ and included a specific date in the YYYY-MM-DD format.

```
mp_bio_1995 <- get_mp(selection="date", fact="bio", date_at ="1995-12-01")
mp_bio_1995
```

When utilizing the get_work function to access data about parliamentary activities, defining a specific time frame becomes essential. As an illustration, consider a scenario where we intend to query and

retrieve details about all parliamentary initiatives, such as bills and resolutions discussed in 2021. Akin to the approach in the get_mp function, you can select a 'fact' parameter to retrieve distinct categories of parliamentary work, either from plenary sessions, as demonstrated in this example, or from committee sessions. Other options for the 'fact' parameter include oral questions and interpellations, written questions, debates, and committee hearings.

```
pi_work <- get_work(date_range_from="2021-01-01"
                    , date_range_to="2021-12-31"
                    , fact="parliamentary_initiatives"
                    , type= "details"
                    , plen_comm= "plen")
pi_work
```

Here is the resulting data frame – with 490 rows and 76 columns.

```
# A tibble: 6 x 76
  id_verg id_fact journaallijn_id fact_link              type_activiteit type_specifiek
  <chr>     <int>           <int> <chr>                  <chr>           <chr>
1 1584146 1566994         1588195 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van d…
2 1584146 1566983         1588195 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van d…
3 1584146 1569590         1588195 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van p…
4 1584146 1566994         1588603 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van d…
5 1584146 1566983         1588605 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van d…
6 1584146 1569590         1588607 http://ws.vlpar.be/e/op… parlementaire_… Ontwerp van p…
```

Modifying the parameter 'type' to "documents" or "speech" yields the corresponding textual content or spoken discourse associated with each parliamentary activity. This capability proves particularly valuable for subsequent text analysis leveraging large language models and various other applications.

```
pi_work <- get_work(date_range_from="2021-01-01"
                    , date_range_to="2021-01-31"
                    , fact="parliamentary_initiatives"
                    , type= "speech"
                    , plen_comm= "plen")
pi_work
```

Finally, merging data is made easy thanks to the unique IDs incorporated in the data. Once data are stored in your R-environment, you can fully use the functionalities included in the *dplyr* package. More broadly, we recommend loading the tidyverse suite of R packages to manipulate the data and visualize some results.

A more thorough overview of data extraction examples and subsequent data manipulations and visualizations can be found at www.flempar.be. Here, we explain in detail how to get started and delve

into more advanced usages of *flempar*, allowing you to work with nested data (i.e., lists in lists) and extract large amounts of data without trouble.

It is also worth highlighting that we have incorporated a feature that allows you to fetch the raw data directly. In essence, the package furnishes you with unprocessed data, offering an additional layer of flexibility for your analytical pursuits. Comprehensive instructions on utilizing this feature are elaborated upon in our blog posts.

How to contribute

The success of the *flempar* open-source package relies on its use, maintenance, and contributions by the social science research community. We, initiators of *flempar*, intend to expand our blog and incorporate more use cases. We also invite researchers to flag any bugs on our GitHub page (https://github.com/PolscienceAntwerp/flempar), contribute their own use cases to our blog, and notify us of their publications based on data retrieved through the *flempar* package.

Conclusion

The potential of open data initiatives has emerged as a pivotal cornerstone in contemporary democratic governance. The commitment of the Flemish Parliament to these principles finds concrete expression in their open data initiative backed up by the license *Gratis Open Data Licentie Vlaanderen*. In this, the Flemish Parliament follows other political institutions around the globe in making a wealth of data available about individual members of parliament and parliamentary work across various policy areas and types of activities. However, data acquired from API querying processes of the Flemish Parliament's open data comes in XML or JSON format, necessitating significant organization and preprocessing efforts before these data can be effectively utilized for analyses. Our *flempar* R-package has been developed to bridge this gap between the open data initiative of the Flemish Parliament and the data structure needs of the social science research community. Specifically, this note has explored the philosophy, scope, and functionalities of the *flempar* R package. Its ability to directly access Flemish Parliament's API means that data can be retrieved and utilized more efficiently. This translates to a more streamlined research process, where effort can be refocused from data acquisition to data analysis and interpretation.

Our note highlights how the package facilitates the work of academics in the social sciences and enhances the work of journalists, data scientists, and policy professionals. We developed installation guides, data extraction examples, detailed blog posts on how to work with nested data and extract large amounts of data spanning multiple years containing thousands of observations, and troubleshooting pages. The ease of installation, the various examples, and detailed troubleshooting guidelines thus ensure

that users at various levels of expertise can take advantage of *flempar*, enhancing its accessibility and real-world applicability. Notably, the package continues to evolve, with plans for updates to incorporate even more data sources and functionalities, keeping it aligned with technological advances and emerging societal needs.

In summary, *flempar* is a pivotal tool for access to the Flemish Parliament's data. It empowers users to easily access, analyze, and interpret a wealth of information, thus contributing to scholarly advancement on the workings of the Flemish Parliament, more transparent governance, and potentially data-driven decision-making.

References

Aikins, S. K., & Krane, D. (2010). Are public officials obstacles to citizen-centered e-government? An examination of municipal administrators' motivations and actions. *State and Local Government Review*, *42*(2), 87-103.

Cahlikova, T., & Mabillard, V. (2020). Open data and transparency: Opportunities and challenges in the Swiss context. *Public Performance & Management Review*, *43*(3), 662-686.

Göbel, S., & Munzert, S. (2022). The comparative legislators database. *British Journal of Political Science*, *52*(3), 1398-1408.

Rauh, C., & Schwalbach, J. (2020). *The ParlSpeech V2 data set: Full-text corpora of 6.3 million parliamentary speeches in the key legislative chambers of nine representative democracies* (V1; Version 2020). https://doi.org/10.7910/DVN/L4OAKN

Søyland, M. (2023). *stortingscrape: Access Data from the Norwegian Parliament API*. In https://CRAN.R-project.org/package=stortingscrape

Zumbach, D., & Gföhler, B. (2022). *swissparl: Interface to the Webservices of the Swiss Parliament*. In https://CRAN.R-project.org/package=swissparl