Replication Code for "Indicators of the Formation of Precedent at the International Court of Justice"

Daniele Bellutta & Kathleen M. Carley
March 2024

How to Cite

Code

Bellutta, Daniele, & Carley, Kathleen M. (2024). "Replication Code for 'Indicators of the Formation of Precedent at the International Court of Justice'". <https://github.com/CASOS-IDeaS-CMU/ICJ>. Accessed 2024-03-15.

Introduction

This repository contains the computer code used to produce the results in the paper "Indicators of the Formation of Precedent at the International Court of Justice". The code mainly comprises several Python scripts, some of which automatically run R code for statistical analysis, that should be executed in a specific order. Some of these scripts pertain only to the decision analysis or to the judge analysis described in the paper, whereas others are necessary for both analyses. What follows is a description of the software required to run the code, the input data necessary to generate results, the purpose of each script, and the order in which to run the scripts. Some additional notes are also included in order to clarify the operation of the code.

Software Requirements

Please ensure that you have the specified versions of the listed packages, since other versions may result in errors.

Python 3.9.12

- `networkx 2.4`
- `statsmodels 0.13.5`
- `numpy 1.22.2`
- `scipy 1.8.0`
- `pandas 1.5.1`
- `rpy2 3.4.5`
- `matplotlib 3.5.1` (used only for figure generation)

R 4.3.2

- `MASS 7.3-60`
- `glm2 1.2.1`
- `lmtest 0.9-40`
- `sandwich 3.1-0`
- `margins 0.3.26` **(see note below on required modifications)**
- `AER 1.2-10`
- `Metrics 0.1.4`
- `bbmle 1.0.25.1`

Input Data

1. JSON file containing citations between ICJ decisions, available from Alschner and Charlotin [1, 2]
2. Directory of ICJ decision text files, available from WorldCourts [3]

Important Notes

Modifications to R `margins` Package

Before installing and using the `margins` package to calculate the marginal effect sizes for the paper's regression models, the code for the package must first be slightly modifying the `margins_glm.R` file within the `R` directory of the package. Before installing the package, download the package source code, navigate to that file, and comment out (or delete) lines 31 and 32 (`model[["model"]] <- NULL` and `attr(model[["terms"]], ".Environment") <- NULL`). Once that has been done, install the package from the modified source code. **If this modification is not performed correctly, the `margins` package will likely raise an error during execution.**

Modifications of Input Data

Reasoning

The first script to run in this repository parses the JSON data published by Alschner and Charlotin [2] and thereafter removes three citations before saving the other citations to CSV files. These citations were excluded from this study because they introduced cycles into the citation network. More specifically, each of these three instances cited cases that appeared to come *after* the citing case. One of these citations appears to be erroneous, whereas the other two are citations to a case that was ongoing at the time it was cited. In the latter two cases, the citing texts refer to the parties' submissions and to a preliminary judgement in the cited case. These submissions and preliminary judgement do not have their own corresponding nodes in the data set, so the citations instead point to the final judgement in the case. To avoid modifying the data too heavily and possibly introducing other errors, these three citations were simply excluded from the study.

Excluded Citations

Link ID (Citation ID)	Source Name (Source Year)	Source ID	Target Name (Target Year)	Target ID	Possible Explanation
1602 (101662)	*Military and Paramilitary Activities in and against Nicaragua (Nicaragua v. United States of America)* (1984)	52	*Aerial Incident of 10 August 1999 (Pakistan v. India)* (2000)	39	The citing text refers to "the Aerial Incident cases" and may therefore have been referring instead to the *Aerial Incident of 27 July 1955 (Israel v. Bulgaria)* case and possibly to the *Appeal Relating to the Jurisdiction of the*

					ICAO Council (India v. Pakistan) case.
2008 (102079)	*Northern Cameroons (Cameroon v. United Kingdom)* (1963)	79	*South West Africa (Liberia & Ethiopia v. South Africa)* (1966)	78	The *South West Africa* case was still ongoing at the time. The citing text in the *Northern Cameroons* judgement refers to the parties' submissions in the *South West Africa* case (and possibly to preliminary judgements in that case), not to the final judgement in that case.
2009 (102080)	*Northern Cameroons (Cameroon v. United Kingdom)* (1963)	79	*South West Africa (Liberia & Ethiopia v. South Africa)* (1966)	78	The *South West Africa* case was still ongoing at the time. This citing text refers specifically to "the Judgment of the Court of 21 December 1962 in the South West Africa cases", which was not the final judgement in the *South West Africa* case and therefore does not have a corresponding node in the data set.

Judge Agreement Networks & Influence Measures

The research paper refers to two agreement networks between judges: the *direct* agreement network and the *direct and indirect* agreement network. The direct and indirect agreement network adds connections between judges who did not serve together on the Court but voted similarly on decisions that were related to each other via citations. In the paper, this network is symmetric, since it connects both newer judges to older ones and older judges to newer ones.

However, the original code for this work also contains the ability to consider indirect agreement in a purely retroactive fashion, in which newer judges are connected to older judges but not *vice versa*. This yields an **asymmetric** indirect agreement network that when added to the direct agreement network, produces yet another asymmetric network.

Because of these different possible definitions, the code in this repository uses the term *indirect agreement* to refer to the **asymmetric** definition of indirect agreement and instead uses the term *symmetric indirect agreement* to refer to the symmetric definition used in the paper. Furthermore, since the code in this repository is set up to handle both symmetric and asymmetric agreement networks, the regression models refer to hub/authority score and in-/out-degree centrality. Since these measures are equivalent to eigenvectory centrality and degree centrality, respectively, in symmetric networks, the paper (which only covers the symmetric agreement networks) refers to them as eigenvector centrality and degree centrality.

Script Descriptions

`convert_data_to_csv.py`

Purpose

This script parses the Alschner and Charlotin JSON data [2] and pulls out the information relevant to this study. **For this study, the original JSON file was modified to remove three citations that pointed from older decisions to newer ones and were therefore deemed erroneous. Please see the note above for more details.**

Input

- `data/original.json` : JSON file specifying citations between ICJ decisions

Output

- `data/cases.csv` : CSV file containing attributes about ICJ decisions
- `data/citations.csv` : CSV file reproducing the JSON citation data in CSV form, minus the excluded citations (see the note above on data modifications for more details)
- Console output: identifiers for the excluded citations (see the note above on data modifications)
    - This output has been included in the file `data/excluded_citations.txt` within this repository.

`extract_judges.py`

Purpose

This script uses regular expressions to find the names of judges in the texts of ICJ decisions and similarly determines how they voted on the decisions.

Input

- `data/decisions/*.txt` : directory of text files, each containing the text of one decision and named using the number corresponding to that decision in the Alschner and Charlotin citation data

Output

- `data/judges.csv` : CSV file containing attributes about ICJ judges
- `data/authorship.csv` : CSV file denoting how each judge voted on a decision

`create_citation_graph.py`

Purpose

This script creates a single file containing the citation network between decisions with added information on the votes in favor of or against each decision (for calculating unanimity).

Input

- `data/citations.csv` : CSV file reproducing the JSON citation data in CSV form
- `data/cases.csv` : CSV file containing attributes about ICJ decisions
- `data/authorship.csv` : CSV file denoting how each judge voted on a decision

Output

- `data/citation_graph.graphml` : GraphML file storing the complete citation network with added vote information

`compute_precedent_scores.py`

Purpose

This script computes all of the dependent, independent, and control variables needed for the decision regression models specified in the paper.

Input

- `data/authorship.csv` : CSV file denoting how each judge voted on a decision
- `data/citation_graph.graphml` : GraphML file containing the complete citation network with added vote information

Output

- `data/decision_variables.csv` : CSV file containing each decision's citation counts and importance measures calculated for every year since its publication
- `data/final_decision_variables.csv` : CSV file containing each decision's importance measures calculated on the final (i.e., complete) citation network

`run_decision_regression.py`

Purpose

This script fits the decision regression models specified in the paper and computes the corresponding average partial effects.

Input

- `data/decision_variables.csv` : CSV file containing each decision's citation counts and importance measures calculated for every year since its publication

Output

- `data/decision_model_coefficients.csv` : CSV file containing the coefficients, robust standard errors, average partial effects, model RMSDs, and **uncorrected** *p*-values for all of the decision regression models

`create_vote_graph.py`

Purpose

This script creates a single file containing the vote network mapping judges to the decisions they voted on, with positive or negative edge weights indicating how they voted on those decisions.

Input

- `data/authorship.csv` : CSV file denoting how each judge voted on a decision
- `data/cases.csv` : CSV file containing attributes about ICJ decisions
- `data/judges.csv` : CSV file containing attributes about ICJ judges

Output

- `data/vote_graph.graphml` : GraphML file containing the network mapping judges to decisions by their votes

`compute_influence_scores.py`

Purpose

This script computes all of the dependent, independent, and control variables needed for the judge regression models specified in the paper.

Input

- `data/citation_graph.graphml` : GraphML file containing the complete citation network with added vote information
- `data/vote_graph.graphml` : GraphML file containing the network mapping judges to decisions by their votes

Output

- `data/judge_variables/*.csv` : directory of CSV files, each containing the yearly judge influence measures and citation statistics computed on one of the types of agreement networks
- `data/direct_final_judge_variables.csv` : CSV file containing each judge's influence measures calculated on the final (i.e., complete) direct agreement network
- `data/direct_and_symmetric_indirect_final_judge_variables.csv` : CSV file containing each judge's influence measures calculated on the final (i.e., complete) direct and symmetric indirect agreement network

`run_judge_regression.py`

Purpose

This script fits the judge regression models specified in the paper and computes the corresponding average partial effects.

Input

- `data/judge_variables/*.csv` : directory of CSV files, each containing the yearly judge influence measures and citation statistics computed on one of the types of agreement networks

Output

- `data/judge_model_coefficients/*.csv` : directory of CSV files, each containing the coefficients, robust standard errors, average partial effects, model RMSDs, and **uncorrected** *p*-values for the judge regression models created using one of the types of agreement networks

`run_fdr_correction.py`

Purpose

This script applies the Benjamini and Hochberg procedure to all of the *p*-values generated from the decision and judge regression models (both for the coefficients and average partial effects) in order to control for a false discovery rate of five percent.

Input

- `data/decision_model_coefficients.csv` : CSV file containing the coefficients, robust standard errors, average partial effects, model RMSDs, and **uncorrected** *p*-values for all of the decision regression models
- `data/judge_model_coefficients/*.csv` : directory of CSV files, each containing the coefficients, robust standard errors, average partial effects, model RMSDs, and **uncorrected** *p*-values for the judge regression models created using one of the types of agreement networks

Output

- `data/corrected_decision_model_coefficients.csv` : CSV file containing the coefficients, robust standard errors, average partial effects, model RMSDs, uncorrected *p*-values, and **FDR-corrected** *p*-values for all of the decision regression models
- `data/judge_model_coefficients/corrected_*.csv` : CSV files containing the

coefficients, robust standard errors, average partial effects, model RMSDs, uncorrected *p*-values, and **FDR-corrected** *p*-values for the judge regression models, with each file created using one of the types of agreement networks

`run_adf_tests.py`

Purpose

This script runs separate augmented Dickey-Fuller (ADF) tests on the dependent variable time series for each decision and judge in the data in order to determine the optimal lag length for each time series. As explained in the paper, this is not a perfect application of lag length testing to the data, since there are effectively multiple time series considered together in the same regression model and since the lagged dependent variables in the regression models for time periods longer than one year need to be recalculated based on past data rather than simply taking the previous year's value for the dependent variable.

Input

- `data/decision_variables.csv` : CSV file containing each decision's citation counts and importance measures calculated for every year since its publication
- `data/judge_variables/direct_judge_variables.csv` : CSV file containing the yearly judge influence measures and citation statistics computed on the direct agreement network
- `data/judge_variables/direct_and_symmetric_indirect_judge_variables.csv` : CSV file containing the yearly judge influence measures and citation statistics computed on the direct and (symmetric) indirect agreement network

Output

- Console output: the proportions of decision or judge dependent variables that, according to the ADF tests, have optimal lag lengths greater than one
  - This output has been included in the file `data/adf_results.txt` within this repository.

`evaluate_lag_lengths.py`

Purpose

For both the decision and judge analysis, this script fits several additional regression models like those specified in the paper but with multiple lags of the dependent variable. The script then calculates the quasi-AIC (since the models are quasi-Poisson models) for each model and determines which lag length yields the lowest quasi-AIC value.

Input

- `data/decision_variables.csv` : CSV file containing each decision's citation counts and importance measures calculated for every year since its publication
- `data/judge_variables/direct_judge_variables.csv` : CSV file containing the yearly judge influence measures and citation statistics computed on the direct agreement network
- `data/judge_variables/direct_and_symmetric_indirect_judge_variables.csv` : CSV file containing the yearly judge influence measures and citation statistics computed on the direct and (symmetric) indirect agreement network

Output

- Console output: the lag lengths yielding the lowest quasi-AIC, maximum resulting coefficient changes (compared to a lag length of one), and maximum resulting APE changes (compared to a lag length of one) for each of the regression models
    - This output has been included in the file `data/lag_length_results.txt` within this repository.

`plot_judge_timeline.py`

Purpose

This script generates the paper's figure showing the judges' durations of tenure on the Court and their level of voting activity over time.

Input

- `data/cases.csv` : CSV file containing attributes about ICJ decisions
- `data/judges.csv` : CSV file containing attributes about ICJ judges
- `data/authorship.csv` : CSV file denoting how each judge voted on a decision

Output

- `figures/judge_timeline.png` : PNG file containing the timeline of judge tenures

Execution Order

To run a script, navigate to the repository directory and execute `python <script_name>` or `python3 <script_name>` in the command line, replacing `<script_name>` with the file name of the script. Please note that this process may vary based on how Python and other software are installed on your computer.

1. Common Pre-Processing

1. `convert_data_to_csv.py`
2. `extract_judges.py`
3. `create_citation_graph.py`

2. Decision Analysis

1. `compute_precedent_scores.py`
2. `run_decision_regression.py`

3. Judge Analysis

1. `plot_judge_timeline.py` (optional)
2. `create_vote_graph.py`
3. `compute_influence_scores.py`
4. `run_judge_regression.py`

4. Common FDR Correction

1. `run_fdr_correction.py`

5. Common Lag Length Testing

1. `run_adf_tests.py` (optional)
2. `evaluate_lag_lengths.py` (optional)

Intermediate Products

The following intermediate data products have been included in this repository to facilitate running only some parts of the code without having to start from the very beginning.

- `data/excluded_citations.txt` : identifiers for the excluded citations (see the note above on data modifications)
- `data/judges.csv` : CSV file containing attributes about ICJ judges
- `data/authorship.csv` : CSV file denoting how each judge voted on a decision
- `data/decision_variables.csv` : CSV file containing each decision's citation counts and importance measures calculated for every year since its publication
- `data/final_decision_variables.csv` : CSV file containing each decision's importance measures calculated on the final (i.e., complete) citation network
- `data/vote_graph.graphml` : GraphML file containing the network mapping judges to decisions by their votes
- `data/judge_variables/*.csv` : directory of CSV files, each containing the yearly judge influence measures and citation statistics computed on one of the types of agreement networks
- `data/direct_final_judge_variables.csv` : CSV file containing each judge's influence measures calculated on the final (i.e., complete) direct agreement network
- `data/direct_and_symmetric_indirect_final_judge_variables.csv` : CSV file containing each judge's influence measures calculated on the final (i.e., complete) direct and symmetric indirect agreement network

Output Results

The following files contain the paper's main results and have therefore also been included in this repository. Please note that the supplementary information published alongside the paper contains tables for the full regression results. Those tables may be clearer to look at than the CSV files created by these scripts.

- `data/corrected_decision_model_coefficients.csv` : CSV file containing the coefficients, robust standard errors, average partial effects, model RMSDs, uncorrected *p*-values, and **FDR-corrected** *p*-values for all of the decision regression models
- `data/judge_model_coefficients/corrected_*.csv` : CSV files containing the coefficients, robust standard errors, average partial effects, model RMSDs, uncorrected *p*-values, and **FDR-corrected** *p*-values for the judge regression models, with each file created using one of the types of agreement networks

References

1. Alschner, W., & Charlotin, D. (2018a). "The growing complexity of the Inter- national Court of Justice's self-citation network". *European Journal of International Law* 29: 83-112.

doi:10.1093/ejil/chy002.
2. Alschner, W., & Charlotin, D. (2018b). "ICJ network". http://ejil.org/ICJ/. Accessed 2020-05-11.
3. WorldCourts. (2020). "ICJ decisions". http://worldcourts.com/icj/eng/index.htm. Accessed 2020-05-11.

Additional references are available in the paper.

Acknowledgements

This work was supported in part by the Office of Naval Research under grants N000141812106 and N000141712675. The views contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research or the U.S. government.