Replication package for
"A Dynamic Discrete Choice Approach to Attitude Stability and Constraint"
by Alecia Nepaul and Steven Stern

Overview. The code in this replication package constructs the analysis files from the two data sources (Fieldhouse et al., 2024; Davern et al., 2025) using Fortran, Stata, and MATLAB. Multiple executables and scripts are necessary to generate the 5 figures and 9 tables in the main paper, plus the additional 37 exhibits of Supplementary Materials. The replicator should expect all code to run for about 16 hours in total, using the provided initial guesses for the maximization routine.

1 Data Availability Statement

Statement about Rights. I certify that the authors of the manuscript have legitimate access to and permission to use the data used in this manuscript.

Summary of Availability. All data are publicly available.

Details on each Data Source.

Data	Files	Location	Citation
BESIP	`*.prn` (fixed width text format) `*.dta` (Stata format)	`BESIP/data/`	Fieldhouse et al. (2024)
GSS	`*.prn` (fixed width text format) `*.dta` (Stata format)	`GSS/data/`	Davern et al. (2025)

Data for the *British Election Study Internet Panel* were downloaded from `https://www.britishelectionstudy.com/data/` while the *U.S. General Social Survey* is available from `https://gss.norc.org/get-the-data.html` For both applications, the data are included in this replication package in two format, in a fixed width text format for use by Fortran code, in the files with `.prn` extension, and in Stata format, in the `.dta` files, for use with Stata.

2 Computational requirements

Software Requirements. We employed 3 software packages to conduct our analysis:

- The Intel Fortran Compiler, included in Intel's OneAPI HPC Toolkit 2025.3, freely available `https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit-download.html`
- Stata 18, available commercially at `https://www.stata.com/` + SSC packages `estout`, `asdoc`
- MATLAB 2025b, available commercially at `https://www.mathworks.com/`

Controlled Randomness. A random number generator seed can be set in `alecia_parallel.f` by using `iseed=1`, calling `randfn` (a subroutine in the Fortran program) and then, setting `iseed=0` prior to each time `randfn` is called. This method of choosing a seed and of generating subsequent random draws is quite machine specific.

Memory, Runtime, Storage Requirements. The Stata and MATLAB scripts used to generate additional results for standard models in the literature and our plots, respectively, run in seconds on a typical laptop or desktop computer, with negligible memory and storage requirements.

For our main code, in Fortran, execution time is extensive and depends on the machine used. We report two sets of results. On a desktop (Dell Precision workstation, with 32 GB RAM and 20 cores total) the BESIP code runs in roughly 36 hours, while the GSS code takes about 3 hours. On a node of the Cardinal cluster of the Ohio Supercomputer Center (1987), with 96 Xeon cores and 512GB RAM, the BESIP application takes 12 hours and the GSS one rought 45 minutes. These times were obtained by providing the code with an initial guesses for the maximization routine that are close to the eventual solution. Estimating both applications using an uninformed initial guess takes multiple days and likely require multiple restarts, even on the cluster node.

Description of programs/code. We describe first out main Fortran source files and then turn to the auxiliary Stata and MATLAB scripts.

Our main code consists of 7 fixed format Fortran source code, the files with the `.f` extension in both `BESIP/` and `GSS/` folders. The source files are identical across applications, they are included twice for convenience. The main executable is `alecia_parallel.f` while the other 6 file include auxiliary routines and variables. The `datget1` subroutine contains the relevant codebook for the BESIP application, while the `datget3` subroutine includes the equivalent information for the GSS application.

The code is instructed which data files to use for estimation via the `alecia.p` file located in each folder, with backup copies `alecia.p.besip` and `alecia.p.gss`, respectively.

We include GNU Make `Makefile`-s for each application. These should work as-is, if the `ifx` compiler command is available on the system. We also include sample SLURM scripts for job submission to the Ohio Supercomputer Center (1987) cluster, as `cardinal.sh`. Update lines 2 and 5 as needed.

The Fortran code is parallelized using OpenMP, to take advantage of all available cores on a single machine. It writes all output to the standard output stream, which must be redirected to a `log.txt` file manually, for example `./alecia_parallel &> log.txt` GNU Bash scripts, `besip_extract.sh` and `gss_extract.sh`, respectively, parse the `log.txt` file and generate multiple, smaller `.txt` files containing key tables and estimates.

Recoding of the raw primary data is done in Stata, using the `data_prep.do` script, located in `BESIP/data/recode/` and `GSS/data/recode/`, respectively. The codebook for the recoded data is named `codebook.pdf` and is available in in `BESIP/data/` and `GSS/data/`.

A Stata script for each application, in `BESIP/data/` or `GSS/data/`, named `comparison_models.do`, estimates the test/retest correlations, ICC, linear panel models with AR(1) errors, and the dynamic panel model with Arellano-Bond estimator. This output is saved in `comparisons.smcl` Stata log.

The correlation heatmaps and IRF plots in the main text are generated using the MATLAB scripts stored in `MATLAB/`. These `.m` scripts output several `.eps` vector graphics files, for inclusion in our manuscript.

3 Instructions to Replicators

The steps required to reproduce the main text and supplemental materials for the BESIP application are as follows. The results for the GSS application are analogous. For both applications, cleaned and formatted data is included in each folder, no additional downloads are necessary. We include optional download and recoding instructions.

1. (Optional) Recode and prep the original raw data by running the `data_prep.do` Stata script in the `BESIP/data/recode/` folder. The output for this operation is saved in `log_recode.smcl` and `log_recode.pdf` The codebook for the recoded data is `BESIP/data/codebook.pdf`
2. Compile the Fortran code using `make` in the `BESIP/` folder. GNU Make will invoke the `ifx` Fortran compiler with the appropriate optimization flags.
3. (Optional) Set the number of threads used during execution, via the `OMP_NUM_THREADS` environmental variables.
4. Execute the main code and redirect all output to a `log.txt` file:

   ```
   ./alecia_parallel &> log.txt
   ```

5. Execute the `./besip_extract.sh` script, which parses the `log.txt` file and generates multiple `besip_*.txt` files with key coefficients and tables. It relies on the GNU Awk utility available on all Unix and Unix-like systems.[1]
6. Use Stata to run the `comparison_models.do` script in the `BESIP/data/` folder, which saves all output in the `comparisons.smcl` Stata log file, and as `.pdf`, for the results of methods previously employed in the literature (test/retest, ICC, etc.)
7. Use MATLAB to run the `heat.m` script in the `MATLAB/Heatmaps/` folder, to generate the correlation heatmap. Run the `besip.m` script in the `MATLAB/IRFs/` folder, to generate the impulse response functions for the BESIP application. The script `MATLAB/plim_example.m` generates Figure A1. The script `MATLAB/party.m` computes the share of variance attributable to Party Id and its Strength.
8. To construct the Wald test statistics reported in Table A1 of the Supplementary Material, change the first line of `alecia.p` to method 6 and rerun the main code. To construct the LM test statistics reported in Table B8, change the method to 5 and rerun the code.

4 List of exhibits

The table on page 4 compiles a list of paper exhibits, tables and figures, including supplemental materials, together with the output files employed in their production, or other comments where applicable.

[1] The ρ coefficients reported by the code require an additional transformation, which maps the unconstrained maximization range $(-\infty, +\infty)$ to the relevant range $(-1, +1)$, using $f(x) = \frac{2e^x}{e^x+1} - 1$. The standard errors need to be transformed via the Delta Method, with $f'(x) = \frac{2e^x}{(e^x+1)^2}$. In this notation, x is the unconstrained coefficient reported by the code and $f(x)$ is the actual, relevant ρ coefficient reported in the text.

Exhibit	Output File, Source, or Comment
Main text	
Table 1	`BESIP/besip_waves.txt`
Figure 1	Subset of Table B1 (Supplemental material)
Table 2	`BESIP/besip_coefficients.txt`
Figure 2	`MATLAB/Heatmaps/heat.m`
Figure 3	`MATLAB/IRFs/besip.m`
Table 3	`BESIP/besip_variance.txt`
Tables 4–5	`BESIP/besip_coefficients.txt`
Table 6	`GSS/gss_coefficients.txt`
Figure 4	`MATLAB/Heatmaps/heat.m`
Figure 5	`MATLAB/IRFs/gss.m`
Table 7	`GSS/gss_variance.txt`
Tables 8–9	`GSS/gss_coefficients.txt`
Supplemental materials	
Table A1	See step 7 in Instructions to Replicators
Figure A1	`MATLAB/plim_example.m`
Table B1	Fieldhouse et al. (2024) and dates compiled by the authors
Table B2	`BESIP/besip_waves.txt`
Table B3	`MATLAB/party.m`
Tables B4–B6	`BESIP/besip_sample_stats.txt`
Table B7	`BESIP/besip_coefficients.txt`
Table B8	See step 7 in Instructions to Replicators
Table B9	`BESIP/besip_coefficients.txt`
Tables B10–B15	`BESIP/comparisons.smcl`
Table B16	`BESIP/comparisons.smcl`
Table B17	`BESIP/comparisons.smcl`
Table B18	`BESIP/comparisons.smcl`
Tables C1–C4	`GSS/gss_sample_stats.txt`
Tables C5–C7	`GSS/gss_coefficients.txt`
Tables C8–C13	`GSS/comparisons.smcl`
Table C14	`GSS/comparisons.smcl`
Table C15	`GSS/comparisons.smcl`
Table D1	Subset of Tables B10–B18
Table D2	Subset of Tables C8–C15
Table D3	Subset of Tables 2, 6, B5–B6, C2–C3

5 Instruction for Uninformed Initial Guesses (Optional)

In this replication package, we included parameter files (`alecia.p.besip` and `alecia.p.gss`) which provide initial guesses that are close to the eventual code solution. If one wishes to redo the estimation using uninformed initial guesses, a more complex iterative procedure is needed. Parameters can also be estimated only a subset at a time, by setting `icoef=1` or `=0` appropriately in the `.p` file. Suppose, for example, that one wishes to update a subset of parameters. The follow steps describe the procedure…

1. For estimation, choose which parameters you want to update. The `icoef` flags are in the same order as the parameter names at the end of the `alecia.p` file. One can look at the Fortran program to see which parameters are of interest to update. The subroutine `parset2` puts the parameter values into the corresponding model parameters (stored in `common/theta`), and one can search the Fortran code to determine the influence of each model parameter possibly with help from the paper (parameter names in the Fortran program match up to names in the paper).
2. For replication, compare the parameter names at the end of output to the full list of parameter names in `alecia.p` Going sequentially, for any name in the output that is also in the `.p` file list, set the corresponding element of `icoef = 0`; for any name in the `.p` file list that is not in the output, set the corresponding element of `icoef = 2` (anything other than `0` works).
3. Some elements in `icoef = 1`. This is because the corresponding element of the parameter vector cannot be identified. Note that the corresponding element of the initial values of the parameters is stored as a number with only 3 significant digits. Since `icoef = 1` has a special meaning (in terms of convention), when you temporarily want to fix a parameter at its present value, set the corresponding element of `icoef` = something other than `0` or `1`, so that you will remember never to change the `icoef = 1` flags back to `0`.
4. Simulation in Estimation: The estimation routine draws errors to simulate the log likelihood function (maximum simulated likelihood estimation). On our machine, one sets a seed by setting `iseed=1`, calling `randfn` (a subroutine in the Fortran program) and then, setting `iseed=0` prior to each time `randfn` is called. This method of choosing a seed and of generating subsequent random draws is quite machine specific.

Note. The Fortran code employs the following convention: after 8 rounds, each consisting of 100 iterations (see `niter = 100` in the source), the program computes an asymptotic covariance matrix and parameter estimates with standard errors. At that point, the program should be restarted with the new values of the parameters typed in, replacing the old initial values. Estimating the BESIP application required 54 such restarts over multiple days. The GSS application required 9, much faster runs, given the smaller size of the GSS data.

References

Davern, M., R. Bautista, J. Freese, P. Herd, and S. L. Morgan (2025). General Social Survey, 1972–2024 [Machine-Readable Data File] / Principal Investigator, Michael Davern; Co-Principal Investigators, Rene Bautista, Jeremy Freese, Pamela Herd, and Stephen L. Morgan. Sponsored by the National Science Foundation. NORC ed. Data accessed from the GSS Data Explorer website.

Fieldhouse, E., J. Green, G. Evans, H. Schmitt, C. Van der Eijk, J. Mellon, and C. Prosser (2024). British Election Study Internet Panel Waves 1–19. *British Election Study*.

Ohio Supercomputer Center (1987). Ohio Supercomputer Center.