**NES Simulation Research: Best Practices Guide (v1.0)**

**Purpose:** To ensure rigor, reproducibility, and systematic progress in the computational modeling and validation of the Normative Executive System (NES) framework.

**Guiding Principles:** Transparency, Iteration, Documentation, Grounding.

**1. Model Versioning & Code Management:**

* **Version Control (Git/GitHub):**
  + **Use Git:** Track all changes to simulation code, analysis scripts, parameter files, and documentation.
  + **Repository:** https://github.com/jessewwright/hegemonikon
  + **Commit Often:** Commit small, logical changes with clear messages describing the update (e.g., "Implemented collapsing bounds in Assent Gate," "Refactored get\_attributes function").
  + **Branching:** Use branches for developing new features or testing significant changes (e.g., feature/raa\_implementation, experiment/dd\_fitting\_v2) before merging into the main branch.
  + **Tagging:** Tag specific code versions used to generate results reported in documents (e.g., v0.1-stroop-fit, v0.2-dd-fit-final).
* **Code Modularity:** Keep code organized into logical functions and potentially separate modules (e.g., nes\_core.py, simulation\_runner.py, analysis\_utils.py). This improves readability and reusability.
* **Clear Naming:** Use descriptive names for files, functions, and variables (e.g., run\_simulation\_stroop, params\_dd\_fit\_final.json).

**2. Simulation Execution & Logging:**

* **Parameter Files:** Store parameter sets used for specific simulations or fitting runs in separate, easily readable files (e.g., JSON, YAML, or simple Python dicts). Name files clearly (e.g., params\_stroop\_fit\_v3.json).
* **Simulation Scripting:** Write scripts that clearly define:
  + Which model version is being run.
  + Which parameter file is being used.
  + The experimental conditions (task, manipulations).
  + The number of trials per condition.
  + The random seed used (for reproducibility).
* **Output Data:**
  + **Raw Trial Data:** Save detailed trial-by-trial output (choice, RT, correctness, key parameter values for that trial, final evidence levels, etc.) to structured data files (e.g., CSV, Parquet). Include unique simulation run IDs and timestamps.
  + **Summary Data:** Generate summary statistics (mean/median RT, accuracy, choice proportions, fit metrics like SSE/AIC/BIC) and save these separately, clearly linked to the raw data and parameters used.
* **Logging:** Include basic logging in scripts to record start/end times, parameters used, number of trials run, any errors encountered, and the location of output files.

**3. Parameter Fitting & Model Comparison:**

* **Document Targets:** Clearly state the target empirical benchmarks (specific values, source literature) being used for fitting.
* **Specify Fitting Procedure:** Detail the objective function (cost function, normalization methods), the optimization algorithm used (Powell, Nelder-Mead, Bayesian method), parameter bounds, initial guesses, and convergence criteria.
* **Report Fit Quality:** Report not just the best-fit parameters but also the final objective function value (e.g., SSE, -LogLikelihood) and standard model comparison metrics (AIC, BIC) if comparing against alternative models.
* **Visualize Fits:** Generate plots showing the model's predictions with best-fit parameters overlaid on the target data (e.g., RT distributions, choice curves).
* **Parameter Recovery (Advanced):** Simulate data *from* the model with known parameters, then try to recover those parameters using the fitting procedure. This checks if the parameters are identifiable and the fitting method is reliable.

**4. Analysis & Interpretation:**

* **Analysis Scripts:** Keep data analysis separate from simulation code. Use scripts (Python with Pandas/Seaborn/Matplotlib, R, etc.) that load the saved output data and generate summary statistics and plots.
* **Link to Theory:** Explicitly connect simulation results back to the NES framework. How do the findings support or challenge specific components or assumptions (Comparator, Assent Gate, w\_n effect, etc.)?
* **Acknowledge Limitations:** Honestly report limitations of the current simulation (e.g., simplifications made, parameter ranges explored, quantitative mismatches).
* **Generate Figures/Tables:** Create clear, well-labeled figures and tables suitable for inclusion in reports or publications. Save figures in high resolution.

**5. Documentation & Reporting:**

* **Simulation Reports:** Maintain comprehensive reports (like the one you've developed) summarizing the goals, methods, results, and analysis for each major simulation study (Stroop, DD, GNG, MD, Fitting). Keep these versioned.
* **Code Documentation:** Add comments to code explaining complex logic. Use docstrings for functions.
* **README Files:** Include README files in code repositories explaining how to set up the environment and run the simulations/analyses.
* **Central Project Log/Journal:** Keep a running log (can be a simple text file or dedicated notes) tracking key decisions, simulation runs performed, major results, problems encountered, and next steps planned. This helps maintain project continuity.

**6. Iteration & Model Refinement:**

* **Systematic Approach:** Base model refinements (e.g., adding RAA logic, tweaking norm decay) on specific simulation results, theoretical considerations, or failed fits.
* **Track Changes:** Clearly document *why* changes were made to the model architecture or parameters between versions (linking back to simulation results or theoretical arguments). Use version control effectively.
* **Re-validate:** After significant model changes, re-run key benchmark simulations to ensure the changes didn't break previously successful replications.

**Applying These Practices:**

* **Right Now:** Ensure your current simulation scripts save raw trial data, use parameter files (even if simple dicts in the script), log basic run info, and use version control for your code and reports. Clearly document the fitting procedure used for Stroop/DD in the report appendix.
* **Moving Forward:** Prioritize saving raw data systematically. Consider adopting PyDDM if custom code becomes unwieldy. Implement parameter recovery if doing serious quantitative fitting. Formalize the project log.