GAME: Student Protocol for Computational Psychology Dissertations

University of Essex Online (UoEO) Computational Psychology Lab — distilled guide

Version: v1.0 (2026)

What this protocol is (and how to use it)

This is a short, practical guide for dissertation students using computational modelling in psychology. It sits alongside the GAME worksheet and the lab-starter GitHub template repo.

The lab workflow you will practise is:

- simulate → fit → compare

You will start with a replication exercise (fork the lab-starter repo and reproduce the demo), then adapt the same workflow to your own dissertation question and dataset.

What modelling adds to a dissertation

- Mechanistic explanation: a process mechanism that could generate the observed pattern.
- Quantitative prediction: trial-by-trial behaviour, not only group means.
- Model comparison: test whether your mechanism beats a simpler baseline.
- Re-use of public datasets: fast, ethical, and reproducible.

The GAME scaffold (one page)

GAME is the lab's project scaffold. You will fill it repeatedly: toy model → real dataset → dissertation plan.

Stage	In one sentence	What you must produce
G — Goals	Define the target phenomenon, the question, and what 'success' would look like.	A clear question, predicted signatures, a baseline to beat, and success criteria.
A — Algorithms	Specify the model as an executable set of variables and update rules.	Model specification (in words + equations/pseudocode), parameters, and simulation plan.
M — Measurements	Map data to model outputs	Data columns, preprocessing

	and define how you will fit and compare models.	rules, likelihood/fit metric, and model comparison plan.
E — Evolution	Track what changed, why it changed, and what you learned.	A short change log, boundary conditions, and an honest limitations section.

G — Goals checklist

Answer these in plain language (then tighten later):

- What behaviour or phenomenon are you explaining (and at what timescale: trial-by-trial, session-by-session)?
- What is the key prediction your mechanism makes (a behavioural 'signature')?
- What's the simplest baseline explanation you must beat (e.g., random choice, biased coin, condition means)?
- What counts as success: better fit and better prediction than the baseline, plus interpretable parameters.
- What constraints matter (ethics, time, dataset availability, computational complexity)?

A — Algorithms checklist

Write down the model so another person could re-build it from your description:

- State variables: what does the model 'carry forward' from one trial to the next (e.g., values, beliefs, evidence)?
- Update rule: how do those states change after each observation (the core mechanism)?
- Choice rule: how does the model turn states into behaviour (e.g., softmax, threshold, rule-with-noise)?
- Parameters: what the free knobs mean psychologically (and sensible bounds).
- Simulation plan: how you will generate synthetic data and what plots you expect to see.

Worked example (the demo you replicate):

- Prediction error δ = reward − expected value
- Update: value ← value + $\alpha \cdot \delta$ (α = learning rate)
- Choice: softmax($\beta \cdot$value) (β = choice consistency / explore–exploit tilt)

M — Measurements checklist

- What columns are required in the dataset (e.g., trial, condition, choice/RT, outcome/reward)?
- How will you compute fit (typically log-likelihood of observed choices or responses)?

- How will you compare models fairly (same data, same preprocessing, clear metric such as AIC)?
- What plots/tables will you save as outputs (minimum: 2 plots + 1 CSV results table)?
- What sanity checks will you run (e.g., parameter recovery on simulated data, data preview before fitting)?

E — Evolution checklist

Use ‘Evolution’ to make your work auditable and defensible:

- Keep a short reproducibility/change log (what changed, when, and why).
- Record key decisions (baseline choice, exclusions/cleaning rules, parameter bounds).
- Note boundary conditions (when the model fails or when behaviour is indistinguishable from baseline).
- Write limitations honestly (and keep claims appropriately modest).

Minimum reproducible deliverable by Week 2 (dissertation readiness gate)

By Week 2, you should be able to show a supervisor a runnable, checkable pipeline:

- A GitHub repo you control (usually a fork of lab-starter) with a clear README and Colab links.
- One ‘Run all’ Colab notebook that completes without manual fixes.
- A dataset saved to data/ (dummy or real).
- At least 2 plots saved to results/figures/.
- At least 1 results table (CSV) saved to results/tables/ (e.g., fitted parameters or model comparison).
- A short reports/reproducibility_log.md describing what ran, what didn’t, and what you changed.

Reproducibility basics (the lab’s non-negotiables)

The lab prioritises rebuildable model descriptions and checkable workflows. A useful rule: if another student can’t reproduce your outputs from your repo, you can’t rely on the result.

Keep it simple and student-proof:

- Pin the environment where possible (requirements file or clear install cell in Colab).
- Fix random seeds when demonstrating results.
- Save outputs (plots + CSV tables) into the repo folder structure.
- Keep notebooks ‘Run all’ clean and avoid brittle, platform-specific hacks.

- Separate what belongs in the write-up (model specification, assumptions, success criteria) from what belongs in the repo (code, data, run steps).

Recommended best-practice reading:

- Miłkowski, M., Hensel, W. M., & Hohol, M. (2018). Reproducibility and replicability in computational neuroscience: A survey and a proposal. Journal of Computational Neuroscience. https://doi.org/10.1007/s10827-018-0702-z

What you submit (replication pack)

This is a replication exercise to prove you can run the lab workflow independently:

- Completed GAME worksheet (at least G and A completed for the demo model).
- Repo link plus commit hash.
- Notebook 01 and 02 saved back to your repo (run-all).
- Outputs saved (plots + CSV results table).
- One-page reproducibility log + limitations.

After replication: turning this into a dissertation project

Typical next steps after you pass the replication pack:

- Swap in a real dataset and repeat the same pipeline (simulate → fit → compare).
- Add one meaningful baseline or one extension (UG), or add a second competing model (Masters).
- Interpret parameters carefully and keep claims aligned to what the model actually tests.
- Document what generalises and what doesn't (boundary conditions).