A Modeling Workflow for Computational Rationality

Suyog Chandramouli, Aini Putkonen, Sebastiaan De Peuter,
Shanshan Zhang, Danqing Shi, Jussi Jokinen,
Andrew Howes, Antti Oulasvirta

May 3, 2023

Abstract

This living document outlines a modeling workflow for computational models of human behavior. The focus is on generative models that predict behavior as a rational consequence of the information processing mechanisms of minds and their contexts – an assumption that is known as 'computational rationality'.

1 What Is a Workflow?

A modeling workflow is a systematic approach for building valid, reliable and useful models. The concept of modeling workflows is not new, a variety of workflows have been suggested previously in cognitive science, statistics, as well as other fields that involve modeling. Workflows in cognitive science often include four broad steps: model specification, model fitting, model checking, and model selection. Each comprises sub-steps, diagnostic checks, and best practices. In machine learning, the workflow steps are different, reflecting the different purpose and techniques involved.

Why follow a workflow, when it may result in a slower development of the model? Because, as a tradeoff, it offers benefits, such as:

1. **Increased quality**: Workflows subject models to several tests. They allow us to check if our modeling assumptions are valid and if the model is working as intended. Models that pass a workflow tend to perform better and their scope is better known.

2. **Increased efficiency**: By following best practices and avoiding common pitfalls, the efficiency of the modeling process is improved.

3. **Clarified contributions**: Workflows call for evaluation of models and their comparison. Results allow you to quantify progress over previous models and attribute to specific modeling ideas.

4. **Transparency**: You will be better able to explain to others what you did and what your model is based on and why. They will be better able to reproduce your work.

Turning the question upside-down, can you afford not to follow a workflow?

But why do computationally rational models need their own workflow? Computationally rational models have demonstrated remarkable success in predicting and explaining human behavior in complex tasks. However, building them is challenging, because methods from different fields are required. Lieder and Griffits (2020) proposed a workflow for resource-rational models:

1. Define the goals of the cognitive system
2. Develop a formal model of the environment to which the system is adapted
3. Make minimal assumptions about cognitive limitations
4. Derive the optimal behavioral function
5. Examine empirical literature to see if the predictions are met
6. If the predictions are off, then iterate

The following workflow expands this by considering parameter fitting, checking, and selection.

Of particular importance to the workflow is an (approximate) optimization step that predicts rational human adaptation by finding one of the best possible policies for behavior. One important set of methods for achieving this step is borrowed from reinforcement learning.

In the following text, a workflow is introduced for computationally rational models that are instantiated with reinforcement learning (Figure 1). These models are generative: they produce behavior – as opposed to merely predicting summary statistics of behavior. Remember that the workflow is suggestive – not prescriptive. Feel free to pick and mix elements however you see fit!

2 Steps Prior to Modeling

Here are preliminary steps that we suggest before moving on to building a new model:

1. Why is a model built? The objective may be for a scientific reason – a desire to advance cognitive science – or for an applied reason – a desire to build computer systems, e.g. that work with and reason about humans. The objective of a model of typing, for example, might be to understand more about the mechanisms of the human motor system, or it may be to understand the impact of keyboard design on typing speed.

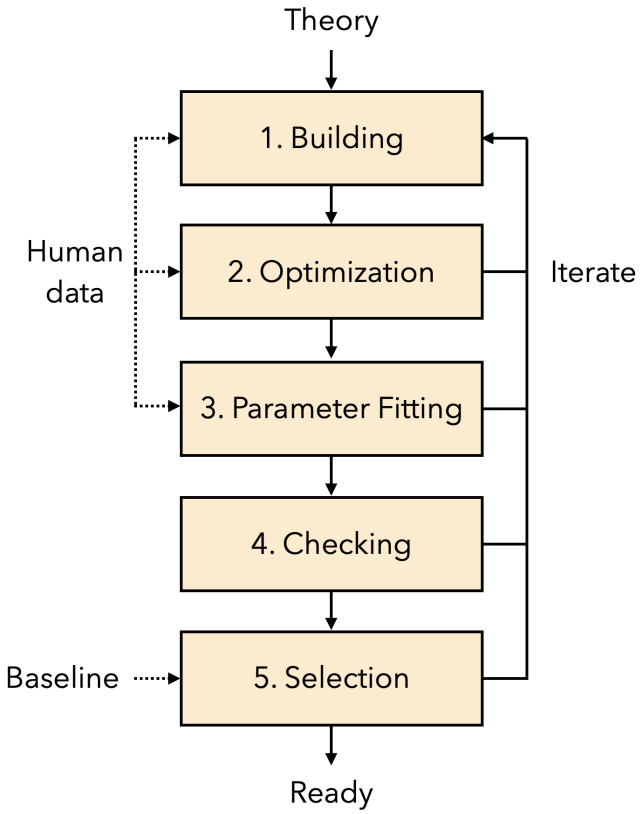

Figure 1: A workflow for developing generative models of human behavior.

2. Defining the scope and goals for the model: By explicitly defining the scope and goals of your model, you will be able to evaluate the eventual model's quality and applicability. This includes defining:

   (a) Theory scope. Which theoretical assumptions do you want to investigate and explicate in the model, and which are secondary and can be learned using machine learning, and which do not belong to the model? Put differently, which components of your theoretically informed generative model are informed by theories of cognition, and which components are informed purely by the task and the data? This involves selection of the cognitive process(es) that are relevant in the context (e.g. attention, memory, perception, decision-making, etc.), and claims we have about them. This forms the broader research question we seek to resolve.

   (b) Task scope. In which conditions should the model be able to operate? This can be informed by the experiment which is the data source: the

nature and number of stimuli presented to participants, the objectives for the participants, the type of responses elicited from them, any states that the participants are expected to be in, etc. The application context in mind for a model would be another determinant of the task scope.

(c) Prediction scope. From the model-free analysis, we would have identified important data patterns to predict, their priority, and the fidelity necessary in their predictions. The overarching goal of modeling can influence the prediction scope. Is your goal of modeling description, explanation, or prediction? If description, your concern is only how accurately the model re-describes observable human behavior; If explanation, the validity of your model's latent assumptions should be taken into account; If prediction, mode perform cross-validation strategy.

(d) Implementation objectives. How large, efficient, extensible, or open must your model be?

3. Exploratory data analysis:

   The goal here is to understand any empirical human data at hand, to both, evaluate the quality of the dataset, and to identify effects and data patterns that must be modelled in order to meet the objectives. This step involves visualizations (scatterplots, violin plots, distribution of data across individuals, etc.).

   Once interesting patterns have been identified and prioritized, it helps to consider candidate models from the literature that are relevant. The best can serve as baselines against which to compare our eventual model. This data analysis can, therefore, also involve exploring data simulated from existing candidate models to understand their predictions.

4. Operationalization of evaluation metrics

   How will you measure success? List down key metrics for all phases of the workflow in a way that aligns with the different objectives. (See below for tips).

5. Identifying your model building strategy:

   (a) Bottom-up: We advise defining a research ladder for the project. It is practically impossible to directly go for the highest-fidelity model; therefore, determine what will be the first model that you build and what must it achieve? We often define ladders with 3-4 steps. Each step defines 1) which signature phenomena must be met and 2) at which fidelity (e.g., trends only or absolute numbers).

   (b) Top-down: Sometimes one may have a strategy of starting from a known model that is performing poorly or overfitting, and simplifying it until there is satisfactory performance.

(c) Mixed: One may also have the strategy of trying both of the above strategies simultaneously and integrate the results of the two at a later point.

For the remaining article we will assume that the reader is following a bottom-up strategy as that encapsulates all of the complexities of going from an abstract theory to a satisfactory model.

3 Build Your Model

Computationally rational models are built around formal descriptions of a sequential decision problem. POMDP stands for a partially observable Markov decision process. The most direct way to get started it to define each element one by one (Figure 2):

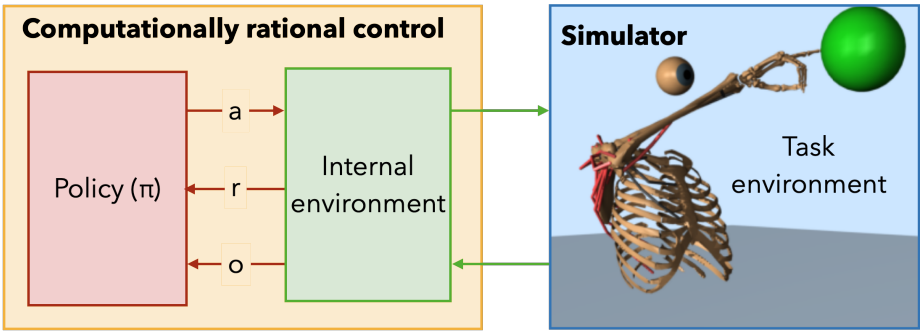

Figure 2: A workflow for developing generative models of human behavior.

1. **The internal environment** is assumed to represent mental states of human-like agents and individually determined bounds. It also transforms stimuli and rewards from the external environment to internal precepts and internal rewards. Start by answering four questions, each demarcated by an arrow in Figure 2: 1) How are stimuli from the simulator processed by cognition? 2) How does cognition form the responses that are sent to the simulator? 3) How are observations (o) of the policy produced by cognition? This is the observation space. 4) What are the actions (a) the control policy sends to cognition? This is called the action space.

2. **The external environment** includes the body of the human, the artefacts, contexts, and interactions among them. You need to decide how the human model's responses affect the environment and how the simulator generates stimuli perceivable by it.

3. **The reward** (r) is defined internally based on beliefs or externally based on events in the simulator. Sometimes we use reward signals generated

by the external environment as a proxy for internal rewards. The key challenge in defining the reward function is to understand how human motivations translate into a mathematical reward function. Your reward variables together define your model's reward space.

4. **The Policy (π)** provides the optimal action to obtain maximum expected reward given the observation. It is also impacted by bounds. More information can be found at section Model Optimization.

Internal Environment: Tips

1. Ultimately you need to answer what state abstractions (or beliefs) does cognition use to condition choice of action? To this end, you need to answer at the following questions:

2. Identify key cognitive bounds that limit the internal environment. For example, are the properties of foveated vision and/or the oculomotor system important for you?

3. Identify key cognitive resources in play. Which memory systems are involved? Does executive control have a role? Attention?

4. If foveated vision is important then how is observation accuracy determined as a function of eccentricity? If oculomotor noise is important then does it matter whether the noise is a function of a radial access?

5. Design the flow of information: which module in the internal environment has access to which and what are the costs of access (e.g., retrieval time or effort)?

6. Manage the sizes of observation and action spaces. These directly affect how rapidly your model converges.

7. Identify latent variables that are of theoretical interest. Study them and their properties: do they have meaningful interpretations within this architecture? Are their ranges meaningful (e.g., no negative values for variables indicating a mental capacity)?

8. Be mindful of variable types. The choice between discrete vs. continuous variables, for example, will have consequences on training efficiency.

External Environment: Tips

1. Choose an appropriate level of abstraction. For example, if you are modeling a high level decision making process then it may not be important to capture the dynamics of oculomotor movement.

2. Ensure that the agent's observations are informative about the state of the physical environment. If you would see what the agent sees, would you be able to complete the task?

3. Instead of building one task instance, treat your external environment as a domain from which instances can be sampled.

Reward Functions: Tips

1. Avoid sparsity and include different types of rewards (see also Rewards Shaping: Tips). Besides intermediate and ultimate (task-determining) rewards, what are some negative rewards involved – events and consequences they would like to avoid? Negative rewards involve penalties for actions that can hinder the agent from achieving its goals. They can also involve costs necessary to achieve a goal, e.g. time costs.

2. Consider the discount factor as part of your reward function. How do people discount immediate vs. long-term gains in your task?

3. Treat a reward function as a hypothesis about what is important for the people you model. Go and seek literature or collect new empirical data to test if this hypothesis holds.

4. Try different strategies for modeling rewards. Unfortunately, empirical specification of reward functions is an open research problem. Sometimes they are derived from instructions given to participants (e.g., "Type as fast as possible"), sometimes from psychological literature ('People are curious"), and sometimes by inferring them from data using inverse RL. Researchers often use external events as a proxy for intrinsic (human) rewards. If you are playing a game, it is a reasonable guess that you may be interested in completing a level. However, this is almost never the full picture.

4 Model Optimization

Parameters in a model can be divided into three classes:

1. Theoretically uninteresting latent parameters. We do not care about the values of these as long as the model works (e.g., neural network weights). → Model optimization

2. Theoretically interesting latent parameters, the value of which impacts model behavior under computational rationality (e.g., a parameter describing a user's working memory capacity). → Parameter fitting

3. Hyperparameters which influence the training or parameter fitting. → Model optimization

Training a reinforcement learning agent involves optimizing its policy. This is done by allowing the agent to interact with an external environment repeatedly, and learn over time, the policy that maximizes its cumulative reward obtained

from the environment. After each learning episode, parameters of the agent's policy or action-value function or reward-function are updated using optimization techniques such as gradient descent. Training stops when a predefined convergence criterion is met. There are several methods to train reinforcement learning agents:

1. Value-based methods: parameters of the value function are updated using temporal difference learning methods such as Q-learning, SARSA, or DQN
2. Policy-based methods: parameters of the policy function are updated using a variety of techniques such as Proximal Policy Optimization (PPO), TRPO, etc.
3. Inverse reinforcement learning (IRL) methods: parameters of the reward function are optimized here using linear, Bayesian, or Maximum entropy -based methods.

Choose an appropriate algorithm for training the policy. Be mindful that besides differences in efficiency, policies have different ways of maintaining state. For example, LSTM-based deep RL methods may be able to do more than your internal environment alone would allow, simply because they keep memory of recent observations.

Cross-validation: Tips

There are many ways to from cross-validation sets: leave-user-out, leave-block-out, leave-task-out, leave-condition-out, etc.

Cross-validation should aim to emulate the prediction task that the model might have to face if deployed. For example, if the goal is to predict behavior of a skilled menu user given observations of a novice user, we should pick leave-block-out.

Careless cross-validation sets leak information. Human behavior is not i.i.d.

Hyperparameter Tuning: Tips

Hyperparameters are of practical value, because they affect learning performance (examples: dropout rate, batch size, number of epochs).

We advise that you always tune hyperparameters. If you do not, your model does not get the chance it deserves. The best option is to use an optimization method like Bayesian optimization that, given sufficient trials, guarantees approximately optimal values. The second-best option is to copy whatever values others have used in prior art. No matter what method you use, you must report the values you used.

Reward Shaping: Tips

Reward shaping is a technique for fighting reward sparsity and thereby facilitate training. Additional 'clues' (shaping) are given to the agent during training that might not be normally present in the immediate reward signal. Tips:

1. Design distance-based reward-shapers. These are a moment-by-moment signal indicating how close one is to a sparse goal. For example, we could reward an agent for moving closer in space to a goal, even if the agent could not observe that.
2. Consider all reward terms when designing a reward-shaping signal. For example, if your reward function has one term for effort and another for reaching a goal, can you find a way to shape both terms?
3. Remember that reward shaping can yield unhuman behaviors, especially when the agent gains privileged access to events humans could not possibly access.

Curriculum Design: Tips

Curriculum design refers to the design of the order in which an agent is exposed to tasks. Careful curriculum design can accelerate learning and help generalize skills.

1. Design curricula that start with simple and end with complex tasks. The first tasks can be dead simple. Can you design a radically simplified (toy) version of your task, perhaps a grid world tasks with only two states?
2. Design curricula that advance with the skills of the agent. Be careful if promoting the agent to the next task before sufficient learning has been demonstrated in the previous task.

5 Parameter Fitting

Some questions to ask before parameter fitting include:

1. What are the theoretical parameters of interests and what prior information do we have about them (their ranges or distributions)? Which parameters to fix, and which ones to vary?
2. Are you fitting to the whole population, a subgroup (as a whole), or to individuals?
3. What does it mean to fit a model to human data? Define your discrepancy function.

General methods for parameter fitting include grid search (when the number of free parameters are small in number), maximum likelihood estimation (e.g. by gradient descent), variational Bayes, etc. where the goal in each is to minimize the discrepency function (e.g. log-likelihood, root-mean squared error, etc.)

Parameter Fitting: Tips

1. Use algorithmic approaches that guarantee optimality, such as Approximate Bayesian Computation, Bayesian Optimization, or Nelder-Mead. These methods generally produce better results than manual tuning of parameters.

2. Examine the posterior distributions of the parameters to understand whether the model is ill-defined and if there are alternative explanations.

3. We encourage using summary statistics primarily (e.g., avg. task completion time) and trajectory-based functions (e.g., sequences of fixation points) secondarily. Use histograms and other functions that capture variability in behavior and use KL divergence to compare distributions.

6 Check and Validate

There are several reasons why model may be invalid or inaccurate: (i) incorrect or incomplete data, (ii) flawed modeling assumptions, (iii) inappropriate model structure, (iv) coding errors, (v) overfitting or underfitting to the data. Model checking refers to the qualitative and quantitative procedures that aid in checking the validity of models we build.

1. Visualize model behavior. Informative visualizations plot model predictions against the actual data. These can be done at the level of individuals as well as the level of groups.

2. Visualize variability. Whenever possible, do several runs with the model and plot confidence intervals or other indicators of variability (e.g. Bayesian HDIs) to your plots.

3. Check parameter recovery. This involves generating data from a known parameterization of a given model, and thereafter checking if the model-fitting procedure recovers these parameters. It is common to visualize these with recovery plots that plots estimated parameter values against a chosen range of true parameter values. If the simulated parameters are close to the estimated ones, parameter recovery is sufficient. For good quality parameter recovery checks, you'd want to consider different sets of parameter values, and observe if any regions in the parameter space are particularly problematic.

4. Check model recovery: this takes the philosophy of a parameter recovery check further, and the goal here is to check if a model that was used to simulate some fake data is recovered as the best fitting model to this fake data when compared to other candidate models.

5. Check predictive accuracy. This involves assessing how close model predictions are to a benchmark dataset. Usually, predictive accuracy is measured on some summary statistic (e.g., number of incorrect selections, instead of exact trajectories). If you are working with a Bayesian model, a posterior predictive check (PPC) involves generating predictions with the model and parameter posterior and comparing these against existing data.

6. Assess the model's theoretical plausibility. Are the theoretical constructs (e.g., a convolutional net simulating early vision) behaving during execution as your theory suggests?

7. Try ablations. In an ablation study, we exclude a theoretical construct (e.g., Kalman filter -based belief update), and test how this exclusion affects model performance.

8. Look at outliers. Human behavior almost always shows a long tail. Models sometimes do. It is important to understand which causes are producing outliers. Unless you are sure that an outlier may be due to confounds such as participants not understanding the task, etc. it is not a good practice to remove outliers.

7 Model Selection

Model selection refers to comparison of the main model against competing (baseline) models. It is essential to perform model checks before proceeding to model selection because we have no way to identify bad model performance if all the candidate models we consider are inadequate.

The goals of different quantitative model selection metrics is to in some way balance predictive performance with model complexity. The reason for this is to avoid overfitting to training data and improve generalizability of the model. For example, Information criteria methods (e.g., AIC, BIC, DIC) have an additive component to a measure of it that penalizes complexity for example, based on the number of parameters in the model. These tend to be easier to calculate. WAIC, LOOIC (an approximation to Leave-one-out crossvalidation) are based on sampling. Bayes factor uses the ratio of marginal likelihoods and produces an implicit bias towards simplicity.

All model selection methods are based on heuristics and can fail in special scenarios. So it is important to choose appropriate selection metrics for your modeling goals. Treat model selection as a multi-objective problem: Systematically chart all metrics obtained by all models. Analyze the tradeoffs. It is rarely that one model is better than others in all respects.

8 Concluding Summary

In this write-up we have highlighted steps that can guide the reader towards producing useful and valid computational rationality models of cognition. While the focus of this write-up is on generative models that build upon CR theory and instantiated with reinforcement learning models, this is still broadly useful for anyone building cognitive models. More resources on computational modeling workflows in different contexts can be found in excellent reviews that discuss this topic.