

PIPELINES AND THEIR COMPOSITIONS FOR MODELING AND ANALYSIS OF CONTROLLED ONLINE NETWORKED SOCIAL SCIENCE EXPERIMENTS

Vanessa Cedeno-Mieles, Yihui Ren, Zhihao Hu, Xinwei Deng,
Noshir Contractor, Saliya Ekanayake, Joshua M. Epstein, Brian J. Goode, Chris J. Kuhlman, Dustin Machi,
Michael W. Macy, Madhav V. Marathe, Henning H. Mortveit, Naren Ramakrishnan, Parang Saraf, Nathan Self
Biocomplexity Institute of Virginia Tech; Departments of Computer Science and Statistics
Blacksburg, VA 24061, USA

ABSTRACT

There has been significant growth in online social science experiments in order to understand behavior at-scale, with finer-grained data collection. Considerable work is required to perform data analytics for custom experiments. We also seek to perform repeated networked experiments and modeling in an iterative loop. In this work, we design and build four composable and extensible automated software pipelines for (1) data analytics; (2) model property inference; (3) model/simulation; and (4) results analysis and comparisons between experimental data and model predictions. We design a formal data model to which experiments and models must conform, for reasoning about them. Our data model is for scenarios where subjects can repeat actions (from a set) any number of times over the game duration. Because the types of interactions and action sets are flexible, this class of experiments is large. Two case studies, on collective identity and complex contagion, illustrate use of the system.

1 INTRODUCTION

1.1 Background and Motivation

Online controlled networked social experiments (games) are increasingly used to study social behaviors (Kearns et al. 2009; Charness et al. 2014) and explore phenomena such as collective identity, exploration versus exploitation, and diffusion and contagion (Centola 2010; Mason and Watts 2012; Charness et al. 2014). Computational modeling is useful in understanding and reasoning about these behaviors (Fujimoto et al. 2017). Combining experiments and modeling enables each to inform and guide the other. This combined approach has been done in some studies without automation (Ackland and O’Neil 2011), or purely conceptually (van der Zee and Holkenborg 2010).

An iterative experimental and modeling approach requires several classes of operations: (1) design experiments and collect data, (2) conduct experiments, (3) data fusion and integration, (4) analyze experimental data, (5) develop and verify models (data-driven models if models are based on the data), (6) infer model parameters, (7) run simulations, (8) compare experimental data against model output, (9) exercise models beyond the ranges of experimental data, and (10) iterate (this is one ordering; see Figure 1). To improve human productivity, it is advantageous to automate these operations for improved efficiency, reproducibility, and scalability. Although there are software systems that address some of these operations (Rioux et al. 2008; Jo et al. 2016), an automated and extensible system for evaluating social phenomena through iterative experiments and modeling that addresses all of these issues is lacking.

In this work, we present an experiment/modeling/simulation/analysis environment composed of pipelines to study social behaviors. A **pipeline** is a sequence of operations, each of which performs a useful task by taking one or more inputs and producing one or more outputs. A pipeline combines operations in analyst-specified ways. We distinguish our work from *workflows* because, while pipelines have many common

features with workflows, here we do not account for provenance of digital objects under a data management system; this work is in progress. See Related Work. Figure 1 provides an overview of our system that is described further below. We also provide multiple formalisms, for a data model and a dynamical systems computational model, that underpin our environment. See Figure 2. Throughout this work, *experiment* or *game* means having human subjects interact to achieve some objective, while the actions are recorded for later analysis. *Modeling* refers to building mathematical representations (i.e., models) of experiments and *simulation* means running software implementations of models; e.g., of agent-based models (ABMs). For clarity, we purposely avoid ambiguous terms like *computational experiment*.

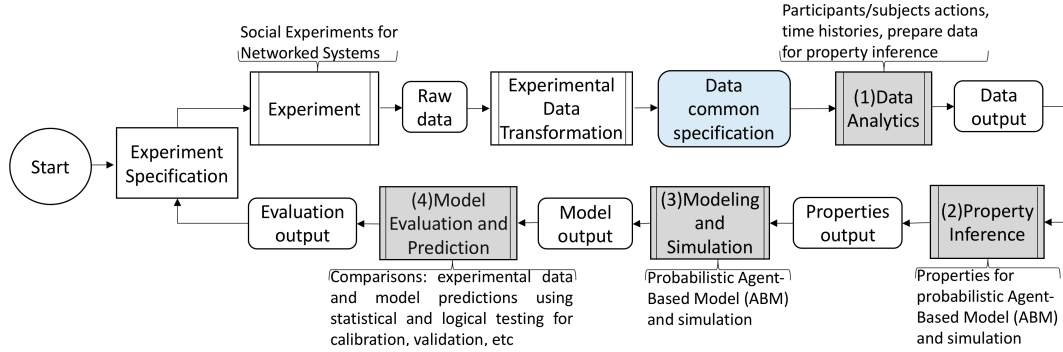


Figure 1: Four pipelines (in gray) for controlled networked online experiments and modeling in the social sciences. In this analysis loop, experiments (experimental platform, upper left) are performed. Experimental data are transformed into a data common specification (in blue) that conforms to our data model (see Section 2). This model provides a single specification that enables any experiment whose data can be cast in terms of the model, to be analyzed in the system (e.g. a classroom experiment). With *controlled* experiments we seek to specify the parameters for a next set of experiments (experiment specification). These pipelines are the focus of this work. This composition of pipelines is one of several possibilities. Our system uses human-in-the-loop.

1.2 Technical Challenges for Social Science Pipelines and Novelty of Work

Technical challenges can be broken down into two categories: those pertaining to pipelines in general, and those that are more specific to social sciences. For large and complex scientific applications, abstractions that capture data processing and computation are important (da Silva et al. 2017). A system is easier to understand and reuse with high-level abstractions (Garijo et al. 2014). General challenges include: identifying the correct levels of abstraction for systems and applications (formal data models and computational models help with this); automation; reproducibility; composability; extensibility; scalability; and traceability (Gil et al. 2007).

For social sciences, we address three challenges, that pertain to social sciences moreso than many fields where pipelines and workflows have been traditionally applied. First, greater range is required in modeling functionality; modeling in social sciences can be different from that in engineering disciplines because often a “model” is a qualitative textual description that is open to different interpretations due to lack of details. Hence a social science “model” can lead to different interpretations and thus resulting algorithmic models to build and evaluate. Second, experiments in the

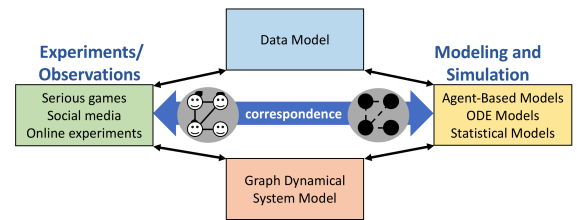


Figure 2: Data (above, blue) and computational (below, red) models enable formal specification of (i) experiments and observations and (ii) modeling and simulation (MAS). They help ensure correspondence between experiments and MAS.

social sciences can vary widely, depending on the phenomena being studied (Thaler 2016). Hence, data analytics, including data exploration, requires custom analyses. Third, different classes of problems require different data and computational models. Melnikov et al. (2018) are pursuing systems similar to that in Figure 1 for quantum studies.

The novelty of this work is devising a data model and applying a computational model that together form abstract representations of experiments and modeling and simulation (MAS) so that we can determine whether an experiment or simulation can be analyzed with our system, and ensure correspondence between experiments and MAS (Figure 2). A second novelty is that our pipelines take a microservices conceptual approach (Cerny et al. 2017) wherein the software components of a pipeline—which we call *functions*—have narrow scopes. Consequently, new functions can be added for new experiments and models in a targeted way, fostering reuse and expanding functionality without introducing redundant capabilities. A third novelty is the range of functions we currently provide: most workflows in the social sciences are for social network analyses (Garijo et al. 2014); we go well beyond that here.

1.3 Contributions

1. Formal data model for networked experiments and simulation modeling. We develop a formal abstract data model for networked social science experiments. The model provides a common representation for both experiments and modeling, thus producing a correspondence between experiments and MAS. It also provides a needed level of abstraction per Section 1.2. Our data model has the following five characteristics: (i) an experiment may be composed of multiple phases (i.e., sub-games); (ii) each phase may have a different finite duration; (iii) each phase may have a different interaction structure among players (i.e., different networks); (iv) each phase may have a different set of actions (and interactions) among players; and (v) these actions may be repeated by players any number of times within the duration of a phase (i.e., temporal interactions). Experiments with our five characteristics represent a significant class of experiments; e.g., (Centola 2010; Mason and Watts 2012; Charness et al. 2014). The data model, with our computational model (Section 3), provides a formal specification for experiments and models. The data common specification in Figure 1 is based on the data model.

2. Four extensible pipelines for modeling and simulation, and analysis, of controlled networked experiments. We design and construct pipelines for (1) data analysis, (2) model property inference, (3) modeling and simulation (MAS), and (4) model evaluation against experiments results, and prediction. Each pipeline consists of an extensible collection of *functions* that can be composed to accomplish computational goals. Moreover, the pipelines themselves can be composed in several ways (Figure 1 is one way). Syntactic data validation of function inputs and outputs ensures robust software execution. The ten operations in Section 1.1 are embedded in these pipelines (software verification is not automated). The Figure 1 caption explains why we emphasize *controlled* experiments; however, use of the pipelines does not require this (e.g., they can be used with social media data). The steps in Figure 1, while automated, are often executed with a human-in-the-loop. The pipelines also satisfy the reproducibility, composability and other “ilities” of Section 1.2.

3. Case studies. We provide two case studies to illustrate use of the system. In one, we describe social experiments that we conducted as a priming activity for collective identity (CI) formation within a group of networked human subjects. All elements in Figure 1 are used. In a second case study, we take the experiment in (Centola 2010) and demonstrate how it maps onto our data model, thus showing that we can evaluate such experiments with our system. Although the number of subjects in experiments is limited to tens of people, our pipelines have been run with millions of artificial subjects for scalability.

Paper Organization. We first present formalisms for the data model (Section 2) and the computational model of discrete dynamical systems (Section 3) because these underpin the pipelines. Next, we present a conceptual view of the pipelines and their components (Section 4), followed by implementation (Section 5). We provide case studies in Section 6, related work in Section 7, and then conclude. A longer unpublished technical report of this work is at (Cedeno et al. 2018).

2 DATA MODEL FOR NETWORKED EXPERIMENTS AND MODELING AND SIMULATION

One of the challenges is to formally represent social experiments and models. Here, we propose a general adaptive abstract data model for networked social experiments with the five characteristics itemized in Contribution 1 of Section 1.3. The purpose of the data model, Table 1, with the computational model of Section 3, is to provide a formal representation for experiments and MAS. Hence, given a description of an experiment or model, one can determine whether our system can be applied; and in reverse, given a phenomenon of study, these models can be used to formulate experiments and models. The “data common specification” in Figure 1 (blue) is produced from this model. For ease of exposition, we describe the data model in terms of an experiment, but it is equally valid for modeling and simulation.

Table 1: Definition of our data model. The experiment schema describes experiment parameters. The phase schema describes parameters for a phase; an experiment can have any number n_p of phases. We use *experiment* throughout in the table and text, but the data model is also used for (*simulation*) models.

#	Parameters	Symbols	Description
Experiment Schema			
1	Experiment id	exp_id	Unique id for an experiment.
2	Number of phases	n_p	Number of phases in an experiment.
3	Number of players	n	The number of unique players over all phases in an experiment.
4	Begin time	t_begin	Timestamp of experiment beginning.
5	End time	t_end	Timestamp of experiment ending.
6	Set of player IDs	V	$V = \{v_1, \dots, v_n\}$. Set of players over all phases; $v_i \in V$ is a unique id for a player.
7	Player attributes	Ω	$\Omega = \bigcup_{j=1}^{n_p} \Omega_j$. $\Omega_j = (\omega_{j1}, \omega_{j2}, \dots, \omega_{j,n_{sa}})$ is the sequence of n_{sa} attributes for $v_j \in V$.
Phase Schema			
1	Phase schema id	ph_sch_id	Unique id for phase schema.
2	Sequence	i_{n_p}	$1 \leq i_{n_p} \leq n_p$. Element of the sequence of phases of the experiment.
3	Phase begin time	t_ph_begin	Timestamp of phase beginning.
4	Phase duration	t_p	Number of time increments in the phase.
5	Unit of time	u_p	Time unit of one time increment (e.g., seconds, days).
6	Network definition	$G(V', E')$	Node set $V' = \{v_1, \dots, v_{\eta}\}$ and edge set $E' = \{e_1, \dots, e_m\}$, where $V' \subseteq V$ may not be all nodes (players) in the system, and $e_i = \{v_j, v_{\ell}\}$ with $v_j, v_{\ell} \in V'$. Note that E' may be empty.
7	Meaning of an edge.	Λ	Set Λ of string representations $\lambda \in \Lambda$ stating the meaning(s) of an edge (e.g., $\lambda = \text{“communication channel”}$ or “influence”).
8	Node attributes for a phase.	Γ	$\Gamma = \bigcup_{t=0}^{t_p} (\bigcup_{j=1}^{\eta} \Gamma_j(t))$. $\Gamma_j(t) = (\gamma_{j1}(t), \gamma_{j2}(t), \dots, \gamma_{j,\eta_v}(t))$ is the sequence of η_v attributes for $v_j \in V'$ in the phase i_{n_p} at time t . $\Gamma_j(t)$ is a sequence of sequences; outer sequence is time.
9	Edge attributes for a phase.	Ψ	$\Psi = \bigcup_{t=0}^{t_p} (\bigcup_{j=1}^m \Psi_j(t))$. $\Psi_j(t) = (\psi_{j1}(t), \psi_{j2}(t), \dots, \psi_{j,\eta_e}(t))$ is the sequence of η_e attributes for $e_j \in E'$ in the phase i_{n_p} at time t . $\Psi_j(t)$ is a sequence of sequences; outer sequence is time.
10	Initial conditions for nodes	B^v	Nodes: $B^v = \bigcup_{j=1}^{\eta} B_j^v$. $B_j^v = (b_{j1}, b_{j2}, \dots, b_{j,\mu_v})$ is the sequence of μ_v initial conditions for the phase, for $v_j \in V'$; $\mu_v \geq 0$.
11	Initial conditions for edges	B^e	Edges: $B^e = \bigcup_{j=1}^m B_j^e$. $B_j^e = (\beta_{j1}, \beta_{j2}, \dots, \beta_{j,\mu_e})$ is the sequence of μ_e initial conditions for the phase, for $e_j \in E'$; $\mu_e \geq 0$.
12	Action set	A	$A = \{a_1, a_2, \dots, a_{n_a}\}$. Set of n_a actions that each player can execute, over time, any number of times, during a phase, where $n_a \geq 0$.
13	Action sequence	T	$T = \bigcup_{t=0}^{t_p} (\bigcup_{k=1}^{\eta} T_k)$. $T_k = (\sigma_i, a_j, v_{\ell_1}, v_{\ell_2}, t_o, py_q)$ is the schema for an <i>action tuple</i> . σ_i is a string that is a unique identifier for an action sequence. Action $a_j \in A$ is initiated by node $v_{\ell_1} \in V'$, and v_{ℓ_2} is the target node of the action, with edge $e = \{v_{\ell_1}, v_{\ell_2}\} \in E'$. $t_o \in \mathbb{R}$ is the time of the action ($0 \leq t_o \leq t_p$); py_q is the payload represented as a JSON schema.

Experiment Schema. Each experiment has a unique id exp_id , a number n_p of phases, a number n of players, a t_begin timestamp for the beginning of the game, a t_end timestamp for the end of the game. Each player has a unique id v_i for identification. A set of players in an experiment is defined by $V = \{v_1, \dots, v_n\}$.

An experiment has n_{sa} player attributes defined for each player. Player attributes Ω are invariant across phases (e.g., age, education level that might be solicited through a survey).

Phase Schema. Each phase schema has a unique id ph_sch_id , the number i_{n_p} of the phase in the sequence of phases, a t_ph_begin timestamp at the beginning of the phase, number t_p of time increments in the phase and the unit of time u_p of one time increment. Each phase represents the interaction structure among players as a network $G(V', E')$ with meanings of edges Λ . Node attributes Γ and edge attributes Ψ over all nodes and edges capture attribute changes in time. Players and edges may have initial conditions B^v and B^e , respectively. An action tuple T_i , which captures pair-wise interactions between players, may be intimately tied to the attribute sequences Γ and Ψ of a phase because action tuples, for example, may cause or be caused by changes in node and edge attributes. In essence, Γ and Ψ can be viewed as sequences of node and edge states. Items 8 through 11 and 13 of the phase schema in Table 1 follow the same basic pattern, to capture features by node or edge, and by time. There are several sequences of values for a particular node or edge j (e.g., Γ_j , Ψ_j , B_j^v , B_j^e , and T_j). Each entry in these sequences can be scalars, sequences, sets, maps, and other structures. Then, these entries are sequenced over time through the union of entries over time, from time 0 through t_p . The exceptions are the initial conditions B_j^v and B_j^e , because by definition, they are specified only at time 0.

3 GRAPH DYNAMICAL SYSTEM MODEL FOR NETWORKED EXPERIMENTS AND MODELING AND SIMULATION

In this section, we present a computational model known as discrete **graph dynamical systems** (GDS). This model formalizes experiments and MAS by capturing the interactions between pairs of players. That is, we use GDS to specify, build, and execute experiments and simulators of experiments (and of other conditions). We use this GDS model because it is correspondent with the data model of Section 2 and because GDSs represent a general model of computation since they can simulate Turing Machines (see e.g., (Barrett et al. 2006)). We also achieve correspondence between experiments and MAS, per Figure 2.

A synchronous **Graph Dynamical System** (GDS) (Mortveit and Reidys 2007) S is specified as $S = (G, F, W)$, where (a) $G(V, E)$, an undirected graph with $|V| = n$, represents the underlying graph of the GDS, with node set V and edge set E , (b) $F = \{f_1, f_2, \dots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local function** associated with node v_i , $1 \leq i \leq n$, and (c) W is the state space, which is the union of the state space W^v for nodes and the state space W^e for edges; i.e., $W = W^v \cup W^e$. Each undirected edge $\{v_i, v_j\} \in E$ can be represented by two directed edges: v_i to v_j , (v_i, v_j) , and (v_j, v_i) .

Each node of G has a state value from W^v . Each edge of G has a state value from W^e . Each function f_i specifies the local interaction between node v_i and its neighbors in G . The inputs to function f_i are the state of v_i , the states of the neighbors of v_i , and the states of the edges outgoing from v_i in G . Function f_i maps each combination of inputs to $s'_i \in W^v$ for v_i , and to $s'_{ij} \in W^e$ for each directed edge $e_{ij} = (v_i, v_j)$. s'_i becomes the next state of node v_i , and s'_{ij} becomes the next state of e_{ij} . These functions are executed in parallel (i.e., synchronously) at each time step t . Node W^v and edge W^e state spaces in the model are represented as (subsets of) the node (Γ) and edge attributes (Ψ), respectively from Table 1. Attributes may have additional parameters that are not part of the node or edge state, such as gender and age. Also, nodes and edges may have *internal* state are used internally for computation, and *external* state that they expose to other graph elements. Action tuples T_k are also part of the state.

GDS captures local interactions on the network G . Player actions, the network, and many other parameters are well-defined. Local functions, however, are not known, and approximations to them must be inferred from data. One of our pipelines performs operations focused on this goal.

4 HIERARCHICAL PIPELINE CONCEPTUAL VIEW

Pipeline Compositions. Our system is composed of all elements of Figure 1, except for the experiment specification. We specifically separate the experimental platform from the pipelines so that the system can

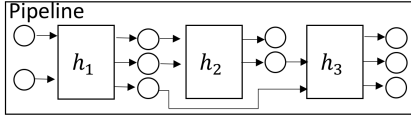


Figure 3: Functions h_i (implemented as software) within a pipeline. Pipelines control the execution order of functions and the inputs and outputs for each function, through a pipeline job specification.

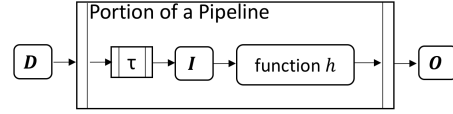


Figure 4: An arbitrary software function h . Input data instances D may have to be transformed τ to conform to required inputs I . Inputs and outputs O are subjected to verification through comparisons with specified schema (not shown).

be used with different experimental software platforms, through the data common specification, via the data model of Section 2. The full system is shown with four pipelines in gray. An iteration of the loop, as shown in Figure 1, may use any number of the four pipelines for flexible composability.

Pipelines. (1) The Data Analytics Pipeline analyzes temporal interactions among players to identify patterns and phenomena in the data. Direct and derived data are used as input for (2) the Property Inference Pipeline. This pipeline generates property values for parameters of simulation models by fusing data from multiple experiments. The simulation models (e.g., agent-based models, ABMs) are built off-line and are part of (3) the Modeling and Simulation Pipeline, which invokes the code to run simulations, using the generated property values, as well as network description, initial conditions, etc. Simulations may model completed or contemplated experiments, or other scenarios beyond the scope of experiments. (4) The Model Evaluation and Prediction Pipeline combines simulation results across multiple (stochastic) executions and performs comparisons between experimental data and model predictions. Note that the pipelines control execution of functionality. Execution control consists of a pipeline invoking **functions** sequentially, as illustrated in Figure 3. Other control structures are being added.

Functions Within Pipelines. Functions are designed as microservices (i.e., modular software with limited scope) within pipelines. Functions h_i ($1 \leq i \leq 3$) in Figure 3 take inputs and generate outputs; these are files, but may include other digital objects, such as database table entries. Figure 4 shows details for a function.

System Formal Description. Due to the space reasons, please see the Formal Pipeline Model Section of our unpublished technical report (Cedeno et al. 2018).

5 PIPELINE IMPLEMENTATION

Pipelines. Pipelines are written in Python. Each takes a configuration file as input. This configuration file states the operations (functions) to execute, the order of execution of these functions, the input and output files for each function, and input and output JSON data schemas for verification of the input and output file formats. A set of feedback parameters (e.g., error codes and pipeline state) are included in a log file of codes (e.g., executing/executed, duration of execution, success or failure, list of inputs and outputs), by function. If necessary, transformation codes transform data into the direct input for a function. The system is programming language agnostic to particular functions. Figure 5 shows the pipeline structure and elements of the execution chain. The pipeline can run on desktops, laptops, and (Linux) clusters.

Functions Within Pipelines. Each pipeline has a list of available functions; Table 2 provides a listing of types of functions h_i within each of the four pipelines. They provide a range of capability from simple plotting routines to cleaning and organizing, storing and accessing data sets, and inferring properties and running simulations. These are not exhaustive; users may add other functions and continue community-based development. Each function completes one well-defined task. Many of these functions can be used in multiple contexts; functions use the pipeline as a universal interface. For example, the action progression function h_3 of the Data Analytics Pipeline generates a plot of the number of actions a_i per player in time

Table 2: Listing of types of functions as microservices within each of the four pipelines. Functions may be considered as collections of functions because they can handle multiple types of data over the data model.

Function - (h_i) Name: Description. Output type and Significance
Pipeline (1): Data Analytics (for experiments and model results)
(h_1) Player interactions : Generate a timeline of individual and between players actions; each player represents a lane; each action has a unique color. Visualization, to detect common patterns between players and actions. (h_2) Timestamp Delta between related actions : Construct a visualization of the timestamp delta between related actions; a request action has a correspondent receive action; each request action represents a lane, a horizontal line represent the length of time it takes to receive a requested action. Visualization, to detect bursts in types of actions. Detect time patterns in types of actions. (h_3) Action progression : Generate a cumulative distribution plot for an action, by player. Data files and plot, to show how an action progresses in time during an experiment phase. (h_4) Average action : Generate plot of the average number of actions between players in a window size s . Data files and plot, to show how an average action progresses in time between experiments phases. (h_5) Action histogram : Generate a histogram of timestamps of an action. Data files and plot, to compare histograms between all experiments phases. (h_6) Histogram of related actions : Generate a histogram of timestamp delta between related actions. Data files and plot, to compare histograms between all experiments phases. (h_7) Discrete action sequence in timeline : Generate a discrete-time action sequence by phase; each action, from the action set A has a unique id definition, with the idle action (i.e., thinking). Time series data files, to generate input for the Property Inference pipeline.
Pipeline (2): Property Inference
(h_1) Properties for Markovian transition matrix : Use of the sequences of discrete actions to generate the probability of transition from an action a_i to an action a_j as measured in the experiment data. Data files, to generates the properties for a Markovian transition matrix. (h_2) Properties for an adapted CRF model : Use of the sequences of discrete actions to generate a derived feature vector accounting for history effects, where the vector corresponds to the discrete-time sequences from the Data Analytics f_7 output. Data files, to generates properties for an adapted conditional random fields (CRF) model. (h_3) Coefficients in a hierarchical model : Generalize the model to take the number of neighbors into consideration, and also digest the additional experiment data where d increases or decreases. Data files, to generate coefficients in a hierarchical model to augment the CRF model. (h_4) Multilinear regression model : Construct multilinear regression model on action set A . Data files, to generate structure of the model and parameter values.
Pipeline (3): Modeling and Simulation
(h_1) Agent based model (ABM) : Execute agent based simulation models; currently, six different models (stationary, dynamic CRF). Data files, to generate Agent Based Model Simulations outputs for self-consistency checks and predictions. (h_2) Statistical regression : Compute a relation between selected and observed values. Data files, to predict most probable value of the observed values for any selected values.
Pipeline (4): Model Evaluation and Prediction
(h_1) Model Validation : Compares experiment outputs with simulation outputs. Data files and plot, to demonstrate that the model is a reasonable representation of the actual system. (h_2) Model Prediction : Generates statistical models to predict outcomes. Data files and plot, to forecast outcomes in an experiment.

$\forall a_i \in A$. Also, often a function represents a category of operation; e.g., there are six different agent-based models (ABMs) under h_1 of the Modeling and Simulation Pipeline. Currently, functions are written in C++, Python, and R. A function may use as input any combination of outputs from preceding functions in the same pipeline, functions in other pipelines, files from previous iterations, and data from experiments.

6 CASE STUDIES

6.1 Study 1: Full System Execution for Collective Identity Experiments

Collective identity (CI) is an individual's cognitive, moral, and emotional connection with an enclosing broader identity such as a team or a community (Polletta and Jasper 2001). A complete game seeks to produce and measure the amount of CI among team players in an experiment. The experiment consists of: phase-1—measure individual levels of CI using the DIFI index (Jiménez et al. 2016) (for a baseline); phase-2—produce CI among team members using a *collaborative* anagrams game; and phase-3—measure individual levels of CI in players using the same index as in phase-1. Due to space constraints, ***we focus on the most complex phase of an experiment***: the phase-2 collaborative anagrams game, motivated by (Charness et al. 2014).

Web-based Experiment Software Platform, Game Play and Data Collection. The primary components of our platform are the oTree framework (Chen et al. 2016), Django Channels and the online web interface. We designed and developed software for each phase of the experiment that interfaces with oTree. Django Channels technology supports interactions among players through websocket communication between

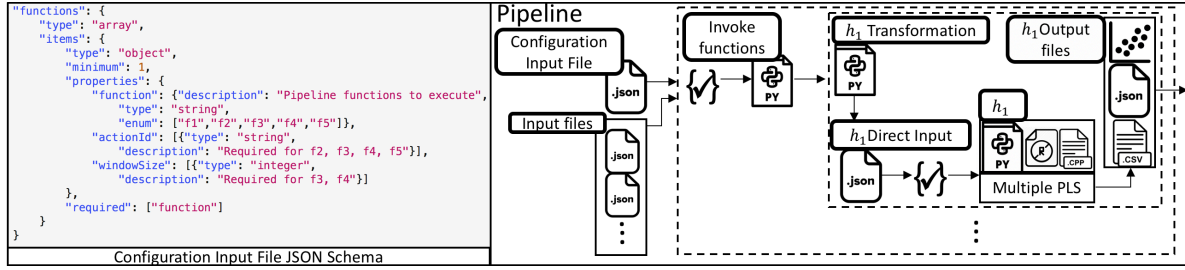


Figure 5: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution (left). At right, we show how function h_1 is executed. Input files are validated against their corresponding JSON schema. If necessary, file contents are transformed (possibly outputs from upstream functions) to obtain the direct inputs for a function in the correct formats. After verification of format by the corresponding JSON schema, the function is executed and output files are generated (these digital object outputs may be, e.g., plot files and ASCII and binary data files).



Figure 6: The anagrams game screen, phase-2, for one player. This player has own letters “R,” “O,” and “L” and has requested an “E” and “A” from neighbors. The “E” is green, so this player’s request has been fulfilled and so “E” can be used in forming words; but the request for “A” is still outstanding so cannot be used in words. Below these letters, it shows that Player 2 has requested “O” and “L” from this player; this player has to reply to these requests, if she so chooses. Below that is a box where the player types and submits new words.

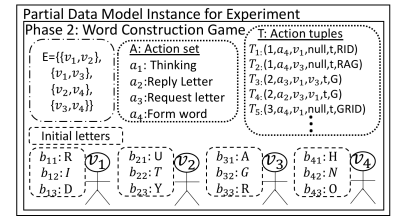


Figure 7: Case study 1. Partial representation of the data model for the online experiment composed of 3 phases with a set of V players ($n = |V|$). Here, we show results for phase 2 anagrams game, a team-based CI-priming game. Edges E are communication channels. Initial conditions B^v include letter assignments to players. The action set A and illustrative action tuples T_i are given for each phase.

individual participants and the server. Figure 6 shows the web interface for each player of the anagrams game. The experimental platform recruits players from Amazon Mechanical Turk and, for all phases of an experiment, records players’ actions. The actions are clicks and their event times for specified HTML objects such as letters, submit buttons, etc.

In phase-2, players are initially given three letters, and are provided communication channels to d number of other players, with whom they can share letters to help each other form words. That is, based on the number n of players recruited, the experimental platform generates a graph on the n players, with a pre-defined regular degree d . Players can form words, request letters from neighbors, and reply to letter requests from neighbors, which are explained in detail in the caption of Figure 6.

The goal is for the *team* to form as many words as possible. Total earnings in this game are based on the total number of words formed by the team, and earnings are split evenly among players. The words

formed by a player have to be unique, but different players can form the same word. Each player has, in effect, an infinite supply of each of her initial three letters so that she can use letters to form words, and also freely share these initial letters with her neighbors. These features are intended to foster cooperation. **Data Analysis, Modeling and Simulations, and Modeling Evaluations using the Pipelines.** Several data model features from Table 1 are provided in Figure 7 for phase-2. The edge set E for the four players is provided, as is the action set A , containing four elements. Initial letter assignments to players, which are part of B_j^v for each node (player) v_j , are shown. So, too, is an illustrative sequence of action tuples.

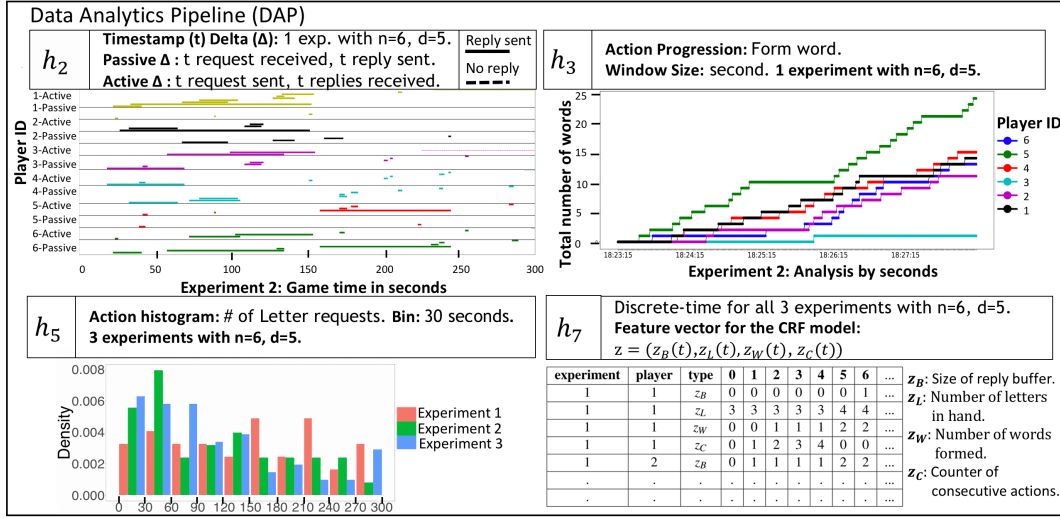


Figure 8: The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_2 of the DAP outputs a visualization for experiment #2. The bars in the active timelines go from the time a request for a letter is sent, until the time that the letter is received. A passive timeline for the same player indicates the time from receiving a letter request until the time this player sends the requested letter. Function h_3 plots the time series of words formed by player for experiment #2. h_5 generates the histogram for the number of actions “letter request” for three experiments. h_7 generates the discrete time actions for all three experiments. This latter output will inform the Property Inference pipeline for computing parameters for simulation models.

Several ABMs were built for this work. The one used here is based on a transition probability matrix where the transition probability from one action $a(t) = a_i$ at time t to the next action $a(t + 1) = a_j$ for each agent v , $i, j \in [1..4]$ and $a(t) \in A$, is given by $\pi_{ij} = Pr(a(t + 1) = j | a(t) = i)$ with $\sum_{j=1}^m \pi_{ij} = 1$. For clarity, we use i and j to represent the actions a_i and $a_j \in A$. Agent v executes a stochastic process driven by transition probability matrix $\Pi = (\pi_{ij})_{m \times m}$, where $m \equiv |A|$ (here, $= 4$). We use a multinomial logistic regression model for π_{ij} . Details are in another paper under review, but also available in Section 4 of the unpublished technical report (Ren et al. 2018). In essence, the ABM predicts action tuples T_i for players v_i in the game, over the 5-minute game duration.

The complete system of Figure 1, and portions of it, were executed over many loops in this study. Here we focus on one iteration of three experiments with $n = 6$ and number of neighbors $d = 5$ to analyze **only the anagrams game**. Figures 8 and 9 show, respectively, results for the Data Analytics and Modeling and Simulation (MAS) and Model Evaluation and Prediction pipelines. Cedeno et al. (2018) provide results for the Property Inference pipeline, too. See the figure captions for details. In this work: (i) output data from the Data Analytics (DA) pipeline are inputs for the Property Inference (PI) pipeline; (ii) outputs from the PI pipeline are inputs to the MAS Pipeline; and (iii) outputs from the DA and MAS pipelines are inputs to the Model Evaluation and Prediction (MEAP) pipeline.

We now address some particular aspects of these results. The plot generated by h_3 in Figure 8 shows, for each player of one game, the time series of words formed. Each step in a curve indicates the time at which a new word is formed. “Form word” is $a_4 \in A$ in Figure 7. The time series for all actions can be formed with h_3 . These data, like those for h_2 and h_5 in Figure 8, are used to (i) understand player behaviors, (ii) assist in specifying the structure of ABMs, (iii) infer properties of ABMs, and (iv) help validate models by comparing model predictions with them. Function h_7 generates the data needed for property inference.

In Figure 9, the Modeling and Simulation Pipeline is used to generate all three plots (the first two for simulating experiments, the third for predictions beyond the experiments). The Model Evaluation and Prediction Pipeline is used in the first two plots to compare experiments and model predictions.

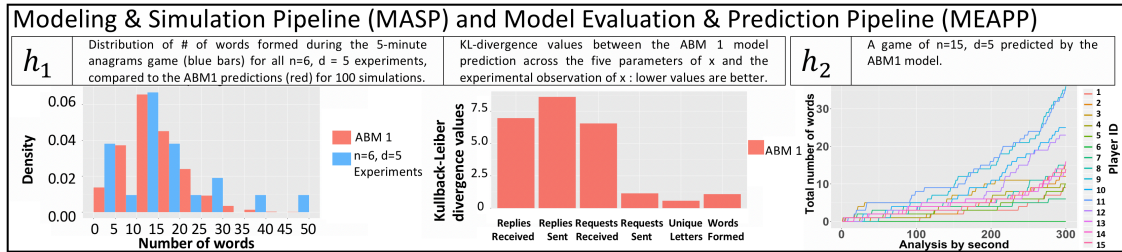


Figure 9: The Modeling and Simulation pipeline (MASP) and Model Evaluation and Prediction pipeline (MEAPP) were executed to generate simulation results and model predictions, and to compare experimental data to model predictions. All three plots use results from h_1 of the MASP. Function h_1 of MEAPP plots corresponding experimental and model output data (left plot) and compares experiment and model output using KL-divergence (center plot) for six parameters. h_3 of the Data Analysis pipeline (DAP) plots model predictions from h_2 of the MASP where now $n = 15$ (in experiments, $n = 6$).

6.2 Study 2: Data Model for Online Experiment in (Centola 2010)

In (Centola 2010), the effects of network structure on complex contagion diffusion are studied by the spread of health behavior through artificially structured online communities. We represent this experiment with the data model from Section 2. Each experiment, exp_id , consists of two independent phases ($n_p = 2$), one with $G(V', E')$ being a clustered-lattice network and another $H(V'', E'')$ being a random network. $V = V' \cup V''$ is the set of all players with player $v_i \in V$, and $1 \leq i \leq n$. There are $n/2$ players in each of the two networks, assigned randomly. Γ_i contains variables for v_i 's profile (i.e., avatar, username, health interests), ratings of the forum content, and the state of v_i in time; i.e., whether v_i has joined the forum. The meaning of an edge is λ = communication channel between pairs of subjects. B_i^v contains initial conditions for the game, including values for the elements of Γ_i . The set of actions is $A = \{a_1, a_2, a_3\}$, where a_1 is “send a message” to encourage a neighbor to adopt a health related behavior; a_2 is “join forum” which also notifies a participant every time a neighbor adopts the behavior; and a_3 is “input rating content” in the forum. Figure 10 shows many of these variables, and examples of action tuples; e.g., T_1 where player 1 sends a message to player 2, and T_6 where player 1 inputs rating content in the forum. This data model instance, coupled with a GDS formulation (not shown), means that the experimental data can be analyzed (and modeled) with the pipeline system. We can perform similar mappings for other social experiments (Kearns et al. 2009; Mason and Watts 2012; Salganik and Watts 2009).

7 RELATED WORK

There are several workflow systems. Examples include Taverna (Wolstencroft et al. 2013) for bioinformatics, chemistry, and astronomy; Pegasus (Deelman et al. 2015) for large-scale workflows in astronomy, seismology, and physics; and Kepler (Barseghian et al. 2010) for ecology and environmental workflows.

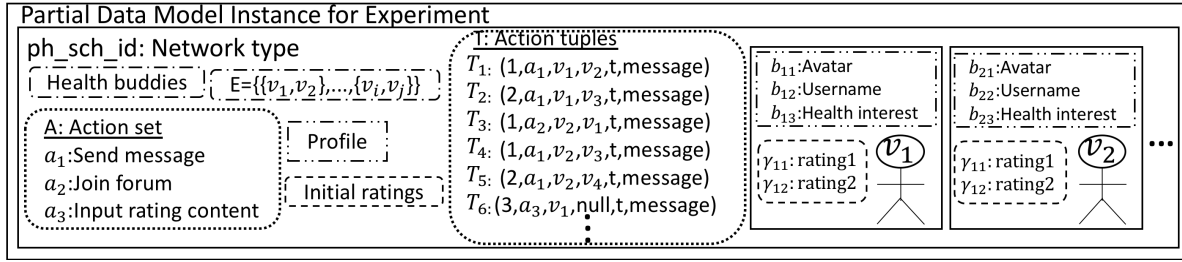


Figure 10: Elements of the data model (Table 1), for the (Centola 2010) online social network experiment.

da Silva et al. (2017) provide an overview of several systems. To the best of our knowledge, none of these systems addresses social sciences for modeling/experiments as we do here. For example, Taverna is used to analyze suicide data in (Sinnott and Hussain 2010); there is no modeling component. Most workflows in the social sciences are for social network analyses (Garijo et al. 2014); we go well beyond that here.

Specialized pipelines are used in different fields. In social sciences, (Jo et al. 2016) develops a three-part pipeline for data analysis and student support in social learning; there is no modeling component. One popular approach is ABM (Macal and North 2009) and simulations. In (Rioux et al. 2008), an XML-based data pipeline for interactive simulation is implemented, but it does not integrate modeling with experiments, nor does it have facilities for comparing model predictions with data. In (van der Zee and Holkenborg 2010) a conceptual model for online games is developed to work with simulations but it does not provide a formal data model for online experiments nor an implementation. (Turner and Lambert 2015) addresses workflows for statistical analysis of social science data. While several of these works address one or a couple of the aspects of our pipelines, none of these works provide formal data models and computational models (e.g., for ABMs) for pipelines, pipeline designs and implementations, pipeline functions, and case studies, as we do.

8 CONCLUSION AND FUTURE WORK

A set of four composable and extensible pipelines for studying networked social science phenomena has been presented, along with data and computational models for formal specification of experiments and MAS, for a particular class of networked social science experiments. Ongoing work includes integrating the pipelines into a distributed data management system, and adding new functions to the pipelines.

REFERENCES

- Ackland, R., and M. O’Neil. 2011. “Online collective identity: The case of the environmental movement”. *Social Networks* 33:177–190.
- Barrett, C. L., H. B. Hunt, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2006. “Complexity of reachability problems for finite discrete dynamical systems”. *J. Comp. Syst. Sci.* 72(8):1317–1345.
- Barseghian, D., I. Altintas, M. B. Jones et al. 2010. “Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis”. *Ecolog. Infor.* 5(1):42–50.
- Cedeno, V., Y. Ren, Z. Hu, X. Deng et al. 2018. “Pipelines and their Compositions for Modeling and Analysis of Controlled Online Networked Social Science experiments”. Technical Report 2018-28. http://ndssl.vbi.vt.edu/supplementary-info/wsc/pipelines_tr.pdf.
- Centola, D. 2010. “The Spread of Behavior in an Online Social Network Experiment”. *Science*:1194–1197.
- Cerny, T., M. J. Donahoo, and M. Trnka. 2017. “Contextual Understanding of Microservice Architecture: Current and Future Directions”. *Applied Computing Review* 17(4):29–45.
- Charness, G., R. Cobo-Reyes, and N. Jimenez. 2014. “Identities, selection, and contributions in a public-goods game”. *Games and Economic Behavior* 87:322–338.
- Chen, D. L., M. Schonger, and C. Wickens. 2016. “oTree—An open-source platform for laboratory, online and field experiments”. *Journal of Behavioral and Experimental Finance* 9:88–97.

- da Silva, R. F., R. Filgueira, I. Pietri, M. Jiang et al. 2017. "A Characterization of Workflow Management Systems for Extreme-Scale Applications". *Future Generation Computer Systems*.
- Deelman, E., K. Vahi, G. Juve, M. Rynge et al. 2015. "Pegasus, a Workflow Management System for Science Automation". *Future Gener. Comput. Syst.* 46:17–35.
- Fujimoto, R. M., C. Carothers, A. Ferscha, D. Jefferson, M. Loper, M. Marathe, and S. J. E. Taylor. 2017. "Computational challenges in modeling simulation of complex systems". In *2017 WSC*, 431–445.
- Garijo, D., P. Alper, K. Belhajjame, O. Corcho, Y. Gil, and C. Goble. 2014. "Common motifs in scientific workflows: An empirical analysis". *Future Generation Computer Systems*:338–351.
- Gil, Y., E. Deelman, M. Ellsman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. 2007. "Examining the Challenges of Scientific Workflows". *IEEE*:24–32.
- Jiménez, J., A. Gomez, M. D. Buhrmester et al. 2016. "The Dynamic Identity Fusion Index: A New Continuous Measure of Identity Fusion for Web-Based Questionnaires". *Soc. Sci. Comp. Rev.*:215–228.
- Jo, Y., G. Tomar, O. Ferschke, C. P. Rosé, and D. Gašević. 2016. "Pipeline for Expediting Learning Analytics and Student Support from Data in Social Learning". In *LAK*, 542–543.
- Kearns, M., S. Judd, J. Tan, and J. Wortman. 2009. "Behavioral experiments on biased voting in networks". *Proceedings of the National Academy of Sciences* 106(5):1347–1352.
- Macal, C. M., and M. J. North. 2009. "Agent-based Modeling and Simulation". In *Winter Simulation Conference*, WSC '09, 86–98.
- Mason, W., and D. J. Watts. 2012. "Collaborative learning in networks". *Proceedings of the National Academy of Sciences* 109(3):764–769.
- Melnikov, A. A., H. P. Nautrup, M. Krenn, V. Dunjko et al. 2018. "Active learning machine learns to create new quantum experiments". *Proceedings of the National Academy of Sciences*.
- Mortveit, H., and C. Reidys. 2007. *An Introduction to Sequential Dynamical Systems*. Springer.
- Polletta, F., and J. M. Jasper. 2001. "Collective Identity and Social Movements". *Ann. Rev. Soc.* 27:283–305.
- Ren, Y., V. Cedeno, Z. Hu, X. Deng et al. 2018. "Generative Modeling of Human Behavior and Social Interactions Using Abductive Analysis". Technical Report 2018-11. http://ndssl.vbi.vt.edu/supplementary-info/wsc/modeling_tr.pdf.
- Rioux, F., F. Bernier, and D. Laurendeau. 2008. "Design and implementation of an XML-based, technology-unified data pipeline for interactive simulation". In *Winter Simulation Conference*, 1130–1138.
- Salganik, M. J., and D. J. Watts. 2009. "Web-Based Experiments for the Study of Collective Social Dynamics in Cultural Markets". *topiCS* 1(3):439–468.
- Sinnott, R. O., and S. Hussain. 2010. "Security-oriented Workflows for the Social Sciences". In *International Conference on Network and System Security*, 152–159.
- Thaler, R. H. 2016. *Misbehaving: The Making of Behavioral Economics*. W. W. Norton & Company.
- Turner, K. J., and P. S. Lambert. 2015. "Workflows for Quantitative Data Analysis in the Social Sciences". *Int. J. Softw. Tools Technol. Transf.* 17:321–338.
- van der Zee, D.-J., and B. Holkenborg. 2010. "Conceptual Modelling for Simulation-based Serious Gaming". In *Winter Simulation Conference*, 522–534.
- Wolstencroft, K., R. Haines, D. Fellows et al. 2013. "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud". *Nucleic Acids Research*:gkt328.

AUTHOR BIOGRAPHIES

VANESSA CEDENO-MIELES, YIHUI REN, ZHIHAO HU, XINWEI DENG, SALIYA EKANAYAKE, BRIAN J. GOODE, CHRIS J. KUHLMAN, DUSTIN MACHI, MADHAV V. MARATHE, HENNING S. MORTVEIT, NAREN RAMAKRISHNAN, PARANG SARAF, NATHAN SELF are with Virginia Tech. Their email addresses are vcedeno@vt.edu, yren2@vt.edu, huzhihao@vt.edu, xdeng@vt.edu, esaliya@vt.edu, bjgoode@vt.edu, ckuhlman@vt.edu, dmachi@vt.edu, mmarathe@vt.edu, hmortvei@vt.edu, naren@vt.edu, parang@vt.edu, nwself@vt.edu .

NOSHIR CONTRACTOR is with Northwestern University. His email address is ncontractor@gmail.com.

JOSHUA M. EPSTEIN is with New York University. His email address is je65@nyu.edu.

MICHAEL W. MACY is with Cornell University. His email address is mwmacy@cornell.edu.