JMLR: Workshop and Conference Proceedings 12 (2011) 115–139 Causality in Time Series

Time Series Analysis with the Causality Workbench

Isabelle Guyon ISABELLE@CLOPINET.COM
ClopiNet, Berkeley, California

Alexander Statnikov ALEXANDER.STATNIKOV@NYUMC.ORG
NYU Langone Medical Center, New York city

Constantin Aliferis CONSTANTIN.ALIFERIS@NYUMC.ORG
NYU Center for Health Informatics and Bioinformatics, New York city

Editor: Florin Popescu and Isabelle Guyon

Abstract

The Causality Workbench project is an environment to test causal discovery algorithms. Via a web portal (`http://clopinet.com/causality`), it provides a number of resources, including a repository of datasets, models, and software packages, and a virtual laboratory allowing users to benchmark causal discovery algorithms by performing virtual experiments to study artificial causal systems. We regularly organize competitions. In this paper, we describe what the platform offers for the analysis of causality in time series analysis.
Keywords: Causality, Benchmark, Challenge, Competition, Time Series Prediction.

1. Introduction

Uncovering cause-effect relationships is central in many aspects of everyday life in both highly industrialized and developing countries: what affects our health, the economy, climate changes, world conflicts, and which actions have beneficial effects? Establishing causality is critical to guiding policy decisions in areas including medicine and pharmacology, epidemiology, climatology, agriculture, economy, sociology, law enforcement, and manufacturing.

One important goal of causal modeling is to predict the consequences of given *actions*, also called *interventions*, *manipulations* or *experiments*. This is fundamentally different from the classical machine learning, statistics, or data mining setting, which focuses on making predictions from observations. Observations imply no manipulation on the system under study whereas actions introduce a disruption in the natural functioning of the system. In the medical domain, this is the distinction made between "diagnosis" and "prognosis" (prediction from observations of diseases or disease evolution) and "treatment" (intervention). For instance, smoking and coughing might be both predictive of respiratory disease and helpful for diagnosis purposes. However, if smoking is a cause and coughing a consequence, acting on the cause (smoking) can change your health status, but not acting on the symptom or consequence (coughing). Thus it is extremely important to distinguish between causes and consequences to predict the result of actions like predicting the effect of forbidding smoking in public places.

The need for assisting policy making while reducing the cost of experimentation and the availability of massive amounts of "observational" data prompted the proliferation of

proposed computational causal discovery techniques (Glymour and Cooper, 1999; Pearl, 2000; Spirtes et al., 2000; Neapolitan, 2003; Koller and Friedman, 2009), but it is fair to say that to this day, they have not been widely adopted by scientists and engineers. Part of the problem is the lack of appropriate evaluation and the demonstration of the usefulness of the methods on a range of pilot applications. To fill this need, we started a project called the "Causality Workbench", which offers the possibility of exposing the research community to challenging causal problems and disseminating newly developed causal discovery technology. In this paper, we outline our setup and methods and the possibilities offered by the Causality Workbench to solve problems of causal inference in time series analysis.

2. Causality in Time Series

Causal discovery is a multi-faceted problem. The definition of causality itself has eluded philosophers of science for centuries, even though the notion of causality is at the core of the scientific endeavor and also a universally accepted and intuitive notion of everyday life. But, the lack of broadly acceptable definitions of causality has not prevented the development of successful and mature mathematical and algorithmic frameworks for inducing causal relationships.

Causal relationships are frequently modeled by causal Bayesian networks or structural equation models (SEM) (Pearl, 2000; Spirtes et al., 2000; Neapolitan, 2003). In the graphical representation of such models, an arrow between two variables $A \rightarrow B$ indicates the direction of a causal relationship: A causes B. A node in the graph corresponding to a particular variable X, represents a "mechanism" to evaluate the value of X, given the "parent" node variable values (immediate antecedents in the graph). For Bayesian networks, such evaluation is carried out by a conditional probability distribution $P(X|Parents(X))$ while for structural equation models it is carried out by a function of the parent variables and a noise model.

Our everyday-life concept of causality is very much linked to time dependencies (causes precede their effects). Hence an intuitive interpretation of an arrow in a causal network representing A causes B is that A preceded B.[1] But, in reality, Bayesian networks are a graphical representation of a factorization of conditional probabilities, hence a pure mathematical construct. The arrows in a "regular" Bayesian network (not a "causal Bayesian network") do not necessarily represent either causal relationships nor precedence, which often creates some confusion. In particular, many machine learning problems are concerned with stationary systems or "cross-sectional studies", which are studies where many samples are drawn at a given point in time. Thus, sometimes the reference to time in Bayesian networks is replaced by the notion of "causal ordering". Causal ordering can be understood as fixing a particular time scale and considering only causes happening at time t and effects happening at time $t + \delta t$, where δt can be made as small as we want. Within this framework, causal relationships may be inferred from data including no explicit reference to time. Causal clues in the absence of temporal information include *conditional indepen-*

1. More precise semantics have been developed. Such semantics assume discrete time point or interval time models and allow for continuous or episodic "occurences" of the values of a variable as well as overlapping or non-overlapping intervals (Aliferis, 1998). Such practical semantics in Bayesian networks allow for abstracted and explicit time.

dencies between variables and loss of information due to *irreversible transformations* or the *corruption of signal by noise* (Sun et al., 2006; Zhang and Hyvärinen, 2009).

In seems reasonable to think that temporal information should resolve many causal relationship ambiguities. Yet, the addition of the time dimension simplifies the problem of inferring causal relationships only to a limited extend. For one, it reduces, but does not eliminate, the problem of confounding: A correlated event A happening in the past of event B cannot be a consequence of B; however it is not necessarily a cause because a previous event C might have been a "common cause" of A and B. Secondly, it opens the door to many subtle modeling questions, including problems arising with modeling the dynamic systems, which may or may not be stationary. One of the charters of our Causality Workbench project is to collect both problems of practical and academic interest to push the envelope of research in inferring causal relationships from time series analysis.

3. A Virtual Laboratory

Methods for learning cause-effect relationships without experimentation (learning from observational data) are attractive because observational data is often available in abundance and experimentation may be costly, unethical, impractical, or even plain impossible. Still, many causal relationships cannot be ascertained without the recourse to experimentation and the use of a mix of observational and experimental data might be more cost effective. We implemented a *Virtual Lab* allowing researchers to perform experiments on artificial systems to infer their causal structure. The design of the platform is such that researchers can submit new artificial systems for others to experiment, experimenters can place queries and get answers, the activity is logged, and registered users have their own virtual lab space. This environment allows researchers to test computational causal discovery algorithms and, in particular, to test whether modeling assumptions made hold in real and simulated data.

We have released a first version `http://www.causality.inf.ethz.ch/workbench.php`. We plan to attach to the virtual lab sizeable realistic simulators such as the Spatiotemporal Epidemiological Modeler (STEM), an epidemiology simulator developed at IBM, now publicly available: `http://www.eclipse.org/stem/`. The virtual lab was put to work in a recent challenge we organized on the problem of "Active Learning" (see `http://clopinet.com/al`). More details on the virtual lab are given in the appendix.

4. A Data Repository

Part of our benchmarking effort is dedicated to collecting problems from diverse application domains. Via the organization of competitions, we have successfully channeled the effort or dozens of researchers to solve new problems of scientific and practical interest and identified effective methods. However, competition without collaboration is sterile. Recently, we have started introducing new dimensions to our effort of research coordination: stimulating creativity, collaborations, and data exchange. We are organizing regular teleconference seminars. We have created a data repository for the Causality Workbench already populated by 15 datasets. All the resources, which are the product of our effort, are freely available on the Internet at `http://clopinet.com/causality`. The repository already includes several time series datasets, illustrating problems of practical and academic interest (see table 1):

- Learning the structure of a fairly complex dynamic system that disobeys equilibration-manipulation commutability, and predicting the effect of manipulations that do not cause instabilities (the MIDS task) (Voortman et al., 2010);
- Learning causal relationships using time series when noise is corrupting data in a way that classical "Granger causality" fails (the NOISE task) (Nolte et al., 2010);
- Uncovering which promotions affect most sales in a marketing database (the PROMO task) (Pellet, 2010);
- Identifying in a manufacturing process (wafer production) faulty steps affecting a performance metric (the SEFTI task) (Tuv, 2010);
- Modeling a biological signalling process (the SIGNET task) (Jenkins, 2010).

The donor of the dataset NOISE (Guido Nolte) received the best dataset award. The reviewers appreciated that the task includes both real data from EEG time series and artificial data modeling EEG. We want to encourage future data donors to move in this direction.

Name (TP; NP; V)	Size	Description	Objective
MIDS (Artificial; NA; 794)	T=12 sampled values in time (unevenly spaced); R=10000 simulations. N=9 variables.	**Mixed Dynamic Systems.** Simulated time-series based on linear Gaussian models with no latent common causes, but with multiple dynamic processes.	Use the training data to build a model able to predict the effects of manipulations on the system in test data.
NOISE (Real + artificial; NA; 783)	**Artificial:** T=6000 time points; R=1000 simul.; N=2 var. **Real:** R=10 subjects. T$\simeq$200000 points sampled at 256Hz. N=19 channels.	**Real and simulated EEG data.** Learning causal relationships using time series when noise is corrupting data causing the classical Granger causality method to fail.	**Artificial task:** find the causal dir. in pairs of var. **Real task:** Find which brain region influence each other.
PROMO (Semi-artificial; 3; 1601)	T=365*3 days; R=1 simulation; N=1000 promotions + 100 products.	**Simulated marketing task.** Daily values of 1000 promotions and 100 product sales for three years incorporating seasonal effects.	Predict a 1000x100 boolean matrix of causal influences of promotions on product sales.
SEFTI (Semi-artificial; NA; 908)	R=4000 manufacturing lots; T=300 async. operations (pair of values {one of N=25 tool IDs, date of proc.}) + cont. target (circuit perf. for each lot).	**Semiconductor manufacturing.** Each wafer undergoes 300 steps each involving one of 25 tools. A regression problem for quality control of end-of-line circuit performance.	Find the tools that are guilty of performance degradation and eventual interactions and influence of time.
SIGNET (Semi-artif.; 2; 2663)	T=21 asynchronous state updates; R=300 pseudodynamic simulations; N=43 rules.	**Abscisic Acid Signaling Network.** Model inspired by a true biological signaling network.	Determine the set of 43 boolean rules that describe the network.

Table 1: **Time dependent datasets.** "TP" is the data type, "NP" the number of participants who returned results and "V" the number of views as of January 2011. The semi-artificial datasets are obtained from simulators of real tasks. N is the number of variables, T is the number of time samples (not necessarily evenly spaced) and R the number of simulations with different initial states or conditions.

5. Benchmarks and Competitions

Our effort has been gaining momentum with the organization of two challenges, which each attracted over 50 participants. The first causality challenge we have organized (Causation and Prediction challenge, December 15 2007 - April 30 2008) allowed researchers both from the causal discovery community and the machine learning community to try their algorithms on sizable tasks of real practical interest in medicine, pharmacology, and sociology (see `http://www.causality.inf.ethz.ch/challenge.php`). The goal was to train models exclusively on observational data, then make predictions of a target variable on data collected after intervention on the system under study were performed. This first challenge reached a number of goals that we had set to ourselves: familiarizing many new researchers and practitioners with causal discovery problems and existing tools to address them, pointing out the limitations of current methods on some particular difficulties, and fostering the development of new algorithms. The results indicated that causal discovery from observational data is not an impossible task, but a very hard one and pointed to the need for further research and benchmarks (Guyon et al., 2008). The Causal Explorer package (Aliferis et al., 2003), which we had made available to the participants and is downloadable as shareware, proved to be competitive and is a good starting point for researchers new to the field. It is a Matlab toolkit supporting "local" causal discovery algorithms, efficient to discover the causal structure around a target variable, even for a large number of variables. The algorithms are based on structure learning from tests of conditional independence, as all the top ranking methods in this first challenge.

The first challenge (Guyon et al., 2008) explored an important problem in causal modeling, but is only one of many possible problem statements. The second challenge (Guyon et al., 2010) called "competition pot-luck" aimed at enlarging the scope of causal discovery algorithm evaluation by inviting members of the community to submit their own problems and/or solve problems proposed by others. The challenge started September 15, 2008 and ended November 20, 2008, see `http://www.causality.inf.ethz.ch/pot-luck.php`. One task proposed by a participant drew a lot of attention: the cause-effect pair task. The problem was to try to determine in pairs of variables (of known causal relationships), which one was the cause of the other. This problem is hard for a lot of algorithms, which rely on the result of conditional independence tests of three or more variables. Yet the winners of the challenge succeeded in unraveling 8/8 correct causal directions (Zhang and Hyvärinen, 2009).

Our planned challenge ExpDeCo (Experimental Design in Causal Discovery) will benchmark methods of experimental design in application to causal modeling. The goal will be to identify effective methods to unravel causal models, requiring a minimum of experimentation, using the Virtual Lab. A budget of virtual cash will be allocated to participants to "buy" the right to observe or manipulate certain variables, manipulations being more expensive that observations. The participants will have to spend their budget optimally to make the best possible predictions on test data. This setup lends itself to incorporating problems of relevance to development projects, in particular in medicine and epidemiology where experimentation is difficult while developing new methodology.

We are planning another challenge called CoMSICo for "Causal Models for System Identification and Control", which is more ambitious in nature because it will perform a continuous

evaluation of causal models rather than separating training and test phase. In contrast with ExpDeCo in which the organizers will provide test data with prescribed manipulations to test the ability of the participants to make predictions of the consequences of actions, in CoMSICo, the participants will be in charge of making their own plan of action (policy) to optimize an overall objective (e.g., improve the life expectancy of a population, improve the GNP, etc.) and they will be judged directly with this objective, on an on-going basis, with no distinction between "training" and "test" data. This challenge will also be via the Virtual Lab. The participants will be given an initial amount of virtual cash, and, as previously, both actions and observations will have a price. New in CoMSICo, virtual cash rewards will be given for achieving good intermediate performance, which the participants will be allowed to re-invest to conduct additional experiments and improve their plan of action (policy). The winner will be the participant ending up with the largest amount of virtual cash.

6. Conclusion

Our program of data exchange and benchmark proposes to challenge the research community with a wide variety of problems from many domains and focuses on realistic settings. Causal discovery is a problem of fundamental and practical interest in many areas of science and technology and there is a need for assisting policy making in all these areas while reducing the costs of data collection and experimentation. Hence, the identification of efficient techniques to solve causal problems will have a widespread impact. By choosing applications from a variety of domains and making connections between disciplines as varied as machine learning, causal discovery, experimental design, decision making, optimization, system identification, and control, we anticipate that there will be a lot of cross-fertilization between different domains.

Acknowledgments

This project is an activity of the Causality Workbench supported by the Pascal network of excellence funded by the European Commission and by the U.S. National Science Foundation under Grant N0. ECCS-0725746. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We are very grateful to all the members of the causality workbench team for their contribution and in particular to our co-founders Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov.

References

C. F. Aliferis, I. Tsamardinos, A. Statnikov, and L.E. Brown. Causal explorer: A probabilistic network learning toolkit for biomedical discovery. In *2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS)*, Las Vegas, Nevada, USA, June 23-26 2003. CSREA Press.

Constantin Aliferis. *A Temporal Representation and Reasoning Model for Medical Decision-Support Systems*. PhD thesis, University of Pittsburgh, 1998.

C. Glymour and G.F. Cooper, editors. *Computation, Causation, and Discovery*. AAAI Press/The MIT Press, Menlo Park, California, Cambridge, Massachusetts, London, England, 1999.

I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov. Design and analysis of the causation and prediction challenge. In *JMLR W&CP*, volume 3, pages 1–33, WCCI2008 workshop on causality, Hong Kong, June 3-4 2008.

I. Guyon, D. Janzing, and B. Schölkopf. Causality: Objectives and assessment. *JMLR W&CP*, 6:1–38, 2010.

Jerry Jenkins. Signet: Boolean rile determination for abscisic acid signaling. In *Causality: objectives and assessment (NIPS 2008)*, volume 6, pages 215–224. JMLR W&CP, 2010.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall series in Artificial Intelligence. Prentice Hall, 2003.

G. Nolte, A. Ziehe, N. Krämer, F. Popescu, and K.-R. Müller. Comparison of granger causality and phase slope index. In *Causality: objectives and assessment (NIPS 2008)*, volume 6, pages 267–276. JMLR W&CP, 2010.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

J.-P. Pellet. Detecting simple causal effects in time series. In *Causality: objectives and assessment (NIPS 2008)*. JMLR W&CP volume 6, supplemental material, 2010.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, Cambridge, Massachusetts, London, England, 2000.

X. Sun, D. Janzing, and B. Schlkopf. Causal inference by choosing graphs with most plausible Markov kernels. In *Ninth International Symposium on Artificial Intelligence and Mathematics*, 2006.

E. Tuv. Pot-luck challenge: Tied. In *Causality: objectives and assessment (NIPS 2008)*. JMLR W&CP volume 6, supplemental material, 2010.

M. Voortman, D. Dash, and M. J. Druzdzel. Learning causal models that make correct manipulation predictions. In *Causality: objectives and assessment (NIPS 2008)*, volume 6, pages 257–266. JMLR W&CP, 2010.

K. Zhang and A. Hyvärinen. Distinguishing causes from effects using nonlinear acyclic causal models. In *Causality: objectives and assessment (NIPS 2008)*, volume 6, pages 157–164. JMLR W&CP, 2009.

Appendix A: Virtual Laboratory

We implemented a virtual laboratory allowing researchers to perform experiments on artificial systems to infer their causal structure. This is part of the "Causality Workbench" effort. The design of the platform is such that:

- Researchers can submit new artificial systems with which other can experiment.
- Experimenters can place queries and get answers.
- The activity is logged.
- Registered users have their own virtual lab space.

We have released a first version **http://www.causality.inf.ethz.ch/workbench.php.**

The canonical way of determining whether events are causally related is to conduct controlled experiments in which the system of interest is "manipulated" to verify hypothetical causal relationships. However, *experimentation* is often costly, infeasible or unethical. This has prompted a lot of recent research on learning causal relationships from available *observational data*. These methods can unravel causal relationships to a certain extent, but must generally be complemented by experimentation.

To this day, the methods, which have emerged in various application domains, have not been systematically compared because of the lack of standard benchmarks. To stimulate research in causal discovery, we created an interactive platform available via a web interface, which will allow researchers to share problems and test methods. Because experimentation is a key component of causal discovery, the platform simulates a laboratory environment in which researchers can experiment via the interface giving access to artificial data generating systems, emulating real systems.

In the virtual laboratory, users can place queries to an unknown system, including request to observe some variables while setting other variables to given values (manipulations), to discover its causal structure. Users have personal accounts and lab space. A given user may choose among a number of data generative systems to reverse engineer. For each new experiment, the user receives: (1) a certain amount of virtual cash to be spent to obtain data from the system by placing queries, (2) information about which variables are observable or actionable (actionable means potentially subject to "action" or "manipulation"), and (3) information about the cost of queries. The system may have hidden variables. The cost of a query varies depending on the number of variables manipulated and/or observed and the number of instances requested. The virtual laboratory keeps track of the queries and virtual cash spent during virtual experiments. When a user runs out of virtual cash, the experiment terminates and he/she must return the answer to the problem. Depending on the nature of the problem, the answer may consist in (i) point-wise predictions of variables (while others are being manipulated), (ii) predictions of distributions, (iii) information on the causal structure of the system, or (iv) a policy to obtain a desired outcome. In this first version, we focus on point-wise predictions of variable values.

Architecture

The system architecture is schematically represented in Figure 1. It is a hub-and-spokes architecture. A central server (the Causality Workbench server) provides a web interface to the users and in equipped with a database.[1] The server communicates with one or several other servers on which causal systems (models) to be studied are simulated. In this first version, we have only one remote server hosting models. This server launches Matlab® upon request of the Causality Workbench server to run the models.

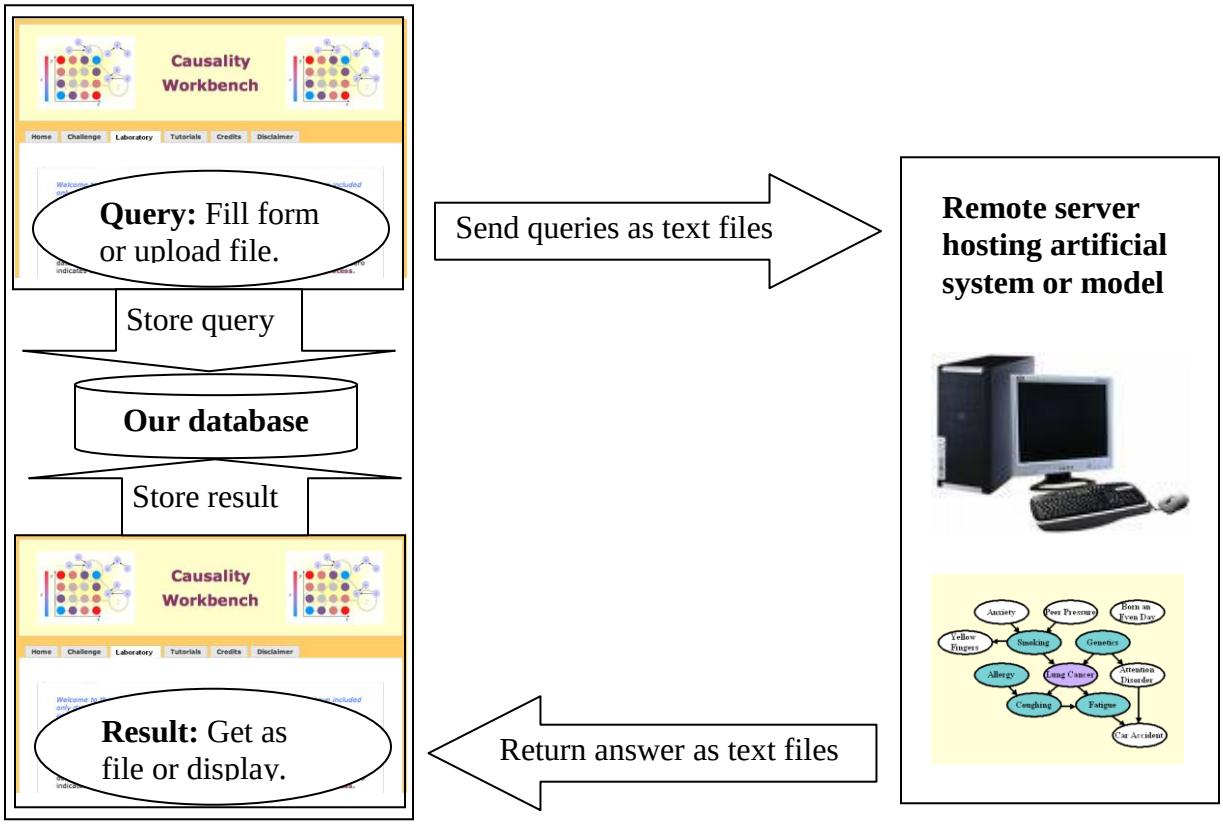


Figure 1: Overview of the Virtual Lab architecture.

Overview of the platform

The Virtual Lab has a web-based user interface (Figure 2) available from http://www.causality.inf.ethz.ch/workbench.php. It is part of the Causality Workbench, which includes other resources (challenges, repository of data/models/software, teleconference seminars, etc.) The Virtual Lab section includes 6 pages. The **Index**, **Leaderboard**, and **Info** pages are publicly available. The **Mylab** and **Upload** pages are only available to registered users.

[1] All implementations are done with PHP and MySQL.

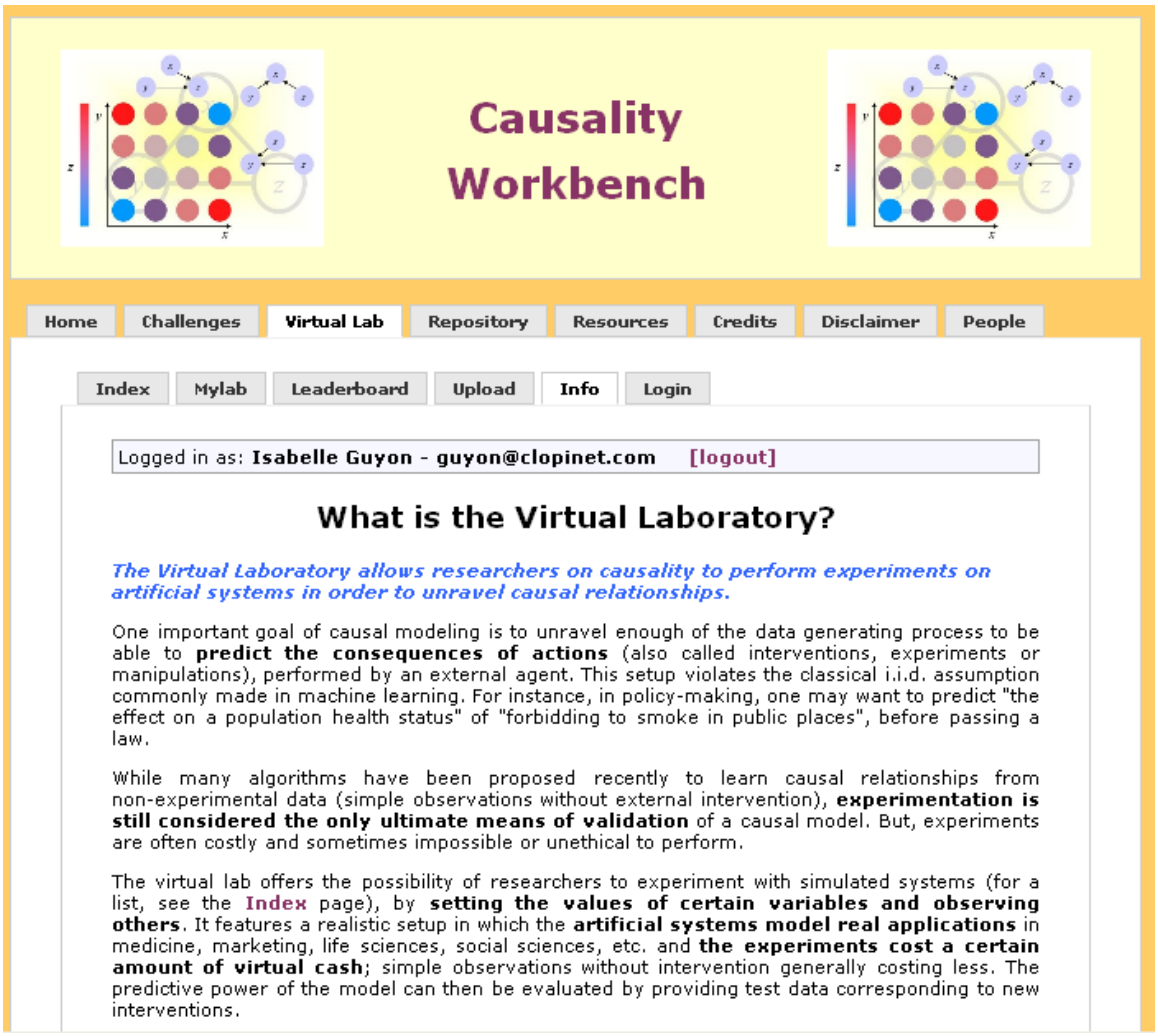
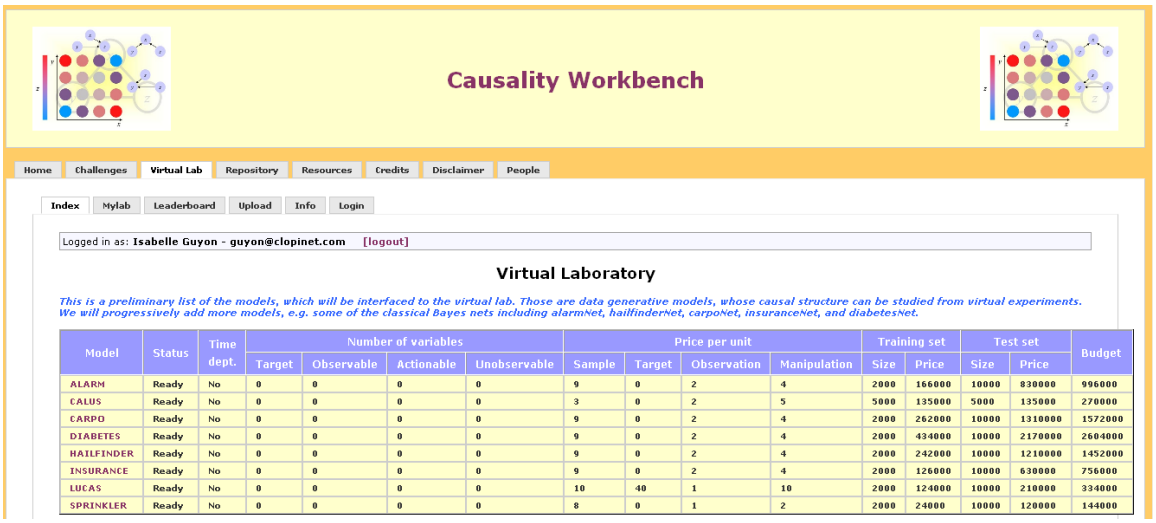


Figure 2: The Virtual Lab Info page.

The users can familiarize themselves with the models, which are listed on the **Index** page (Figure 3). The table lists the model properties and the initial budget (amount of virtual cash available for experimentation) as well as the cost per experimental unit. The models are cross-indexed with the model repository automatically when a new model is registered (Figure 4). At present, model registration is via its inclusion in the GLOP package.



Model	Status	Time dept.	Number of variables				Price per unit				Training set		Test set		Budget
			Target	Observable	Actionable	Unobservable	Sample	Target	Observation	Manipulation	Size	Price	Size	Price	
ALARM	Ready	No	0	0	0	0	9	0	2	4	2000	166000	10000	830000	996000
CALUS	Ready	No	0	0	0	0	3	0	2	5	5000	135000	5000	135000	270000
CARPO	Ready	No	0	0	0	0	9	0	2	4	2000	262000	10000	1310000	1572000
DIABETES	Ready	No	0	0	0	0	9	0	2	4	2000	434000	10000	2170000	2604000
HAILFINDER	Ready	No	0	0	0	0	9	0	2	4	2000	242000	10000	1210000	1452000
INSURANCE	Ready	No	0	0	0	0	9	0	2	4	2000	126000	10000	630000	756000
LUCAS	Ready	No	0	0	0	0	10	40	1	10	2000	124000	10000	210000	334000
SPRINKLER	Ready	No	0	0	0	0	8	0	1	2	2000	24000	10000	120000	144000

Figure 3: The Virtual Lab Index page. The models listed are linked to the repository.

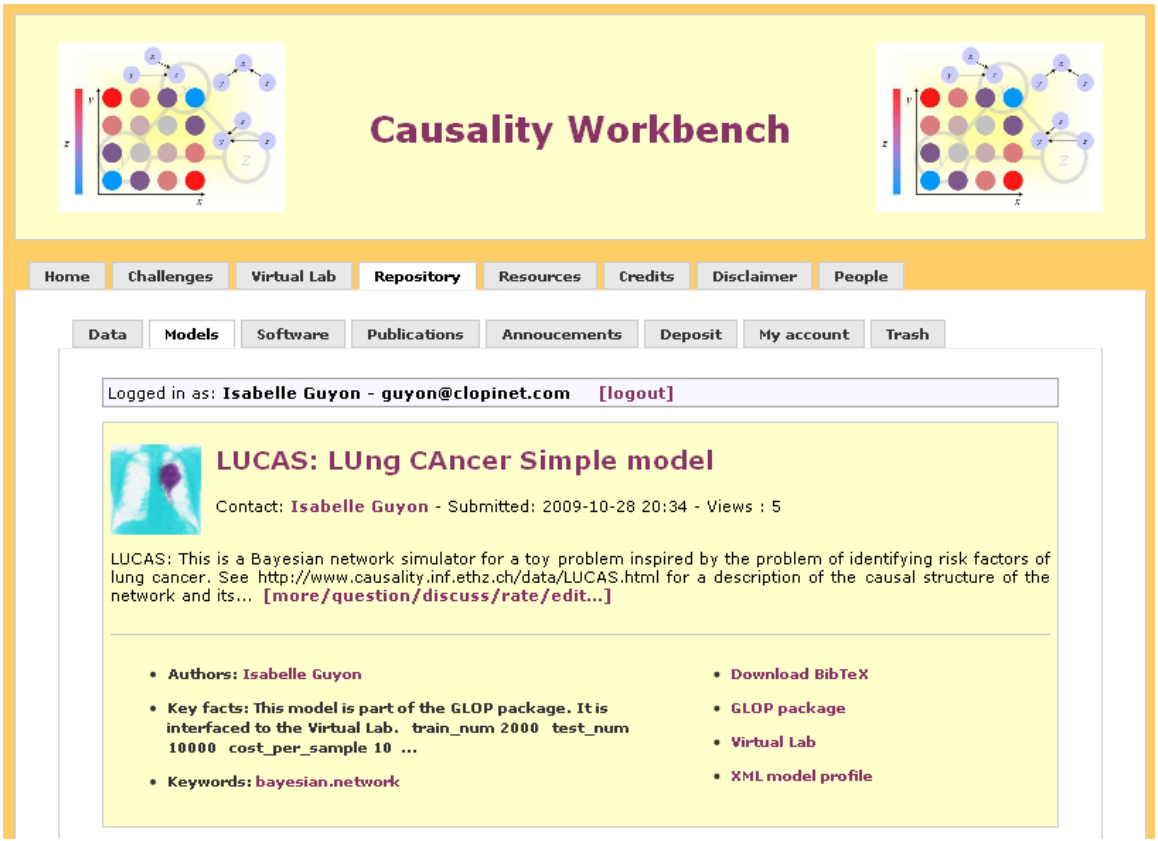
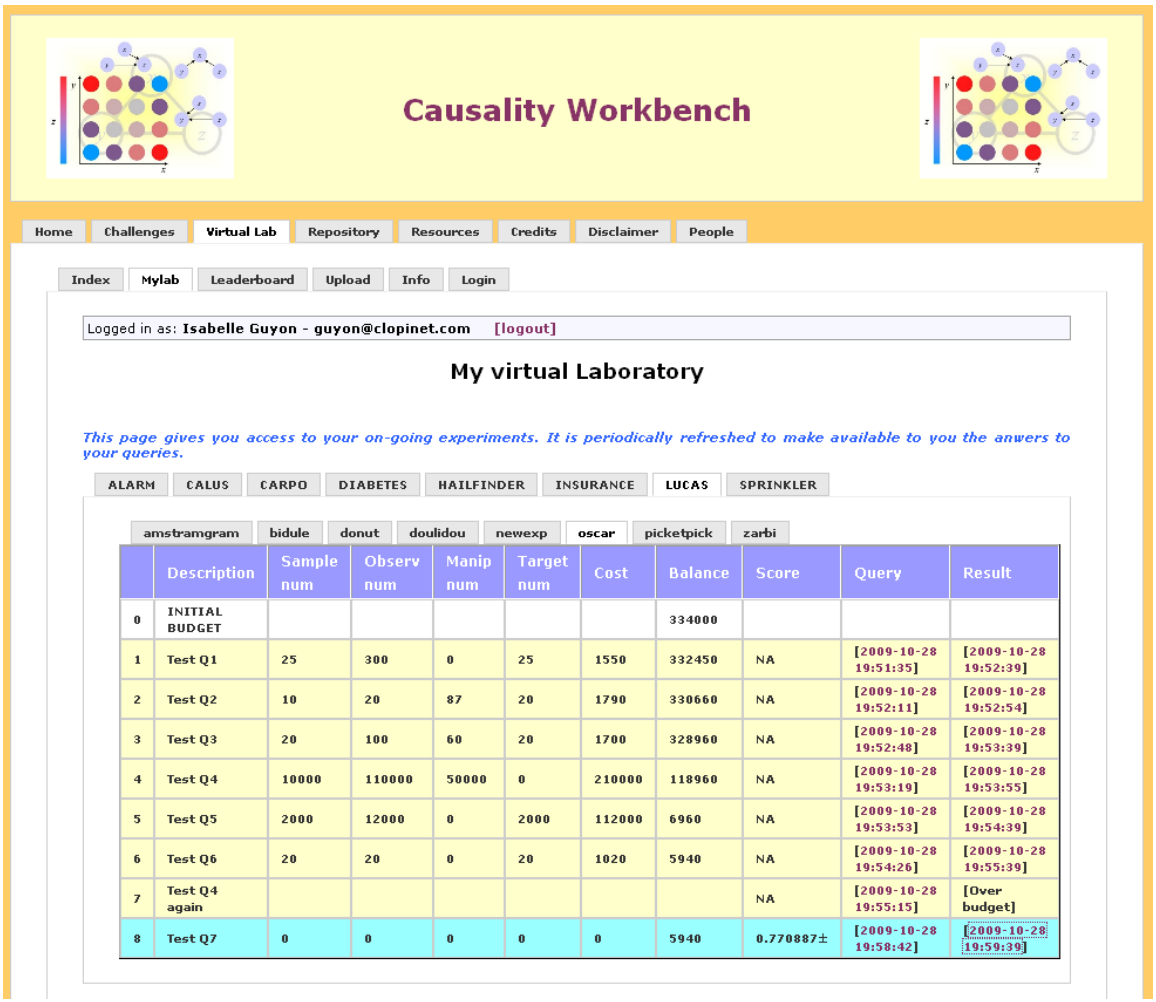


Figure 4: Repository. The registered models are included in the repository.

Following the instructions described in the **Info** page, registered users may upload queries and results from the **Upload** page and retrieve data and performance scores from the **Mylab** page (Figure 5). When final prediction results on test data are uploaded, the experiment terminates and the final score is published on the **Leaderboard** page.



	Description	Sample num	Observ num	Manip num	Target num	Cost	Balance	Score	Query	Result
0	INITIAL BUDGET						334000			
1	Test Q1	25	300	0	25	1550	332450	NA	[2009-10-28 19:51:35]	[2009-10-28 19:52:39]
2	Test Q2	10	20	87	20	1790	330660	NA	[2009-10-28 19:52:11]	[2009-10-28 19:52:54]
3	Test Q3	20	100	60	20	1700	328960	NA	[2009-10-28 19:52:48]	[2009-10-28 19:53:39]
4	Test Q4	10000	110000	50000	0	210000	118960	NA	[2009-10-28 19:53:19]	[2009-10-28 19:53:55]
5	Test Q5	2000	12000	0	2000	112000	6960	NA	[2009-10-28 19:53:53]	[2009-10-28 19:54:39]
6	Test Q6	20	20	0	20	1020	5940	NA	[2009-10-28 19:54:26]	[2009-10-28 19:55:39]
7	Test Q4 again							NA	[2009-10-28 19:55:15]	[Over budget]
8	Test Q7	0	0	0	0	0	5940	0.770887±	[2009-10-28 19:58:42]	[2009-10-28 19:59:39]

Figure 5: Mylab. Each user has his/her personal lab space holding his/her experiments.

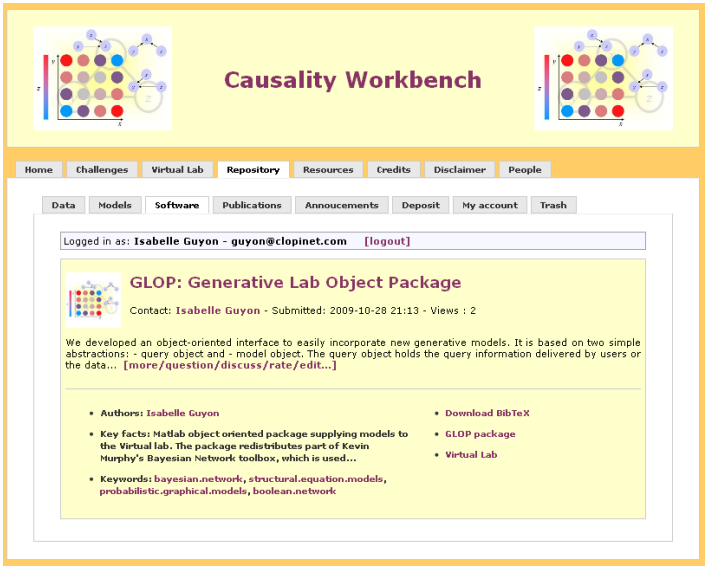


Figure 6: GLOP. The GLOP Matlab package is available for download in the software repository.

The GLOP package

Our first release of the Virtual Lab uses a single remote server running Matlab to implement artificial causal systems. We developed an object-oriented interface to easily incorporate new generative models. It is based on two simple abstractions:

- *query object* and
- *model object*.

The *query object* holds the query information delivered by users or the data delivered by the generative models. It has a fixed structure. The *model object* is a template from which data generative models can be derived. We call GLOP (Generative Lab Object Package) the resulting package of objects. GLOP may be downloaded from http://www.causality.inf.ethz.ch/repository.php?id=23 (Figure 6).

```
%=============================================================
% QUERY object
%=============================================================
% q = query;
% q = query(filename);
% q = query(filename, fn2);
%
% Stores query information and results.
% The argument "filename" is the root name of the query file.
% If fn2 is specified, make a copy to the second file
%
% One expects a subset of the files:
% <filename>.type: Type of the query, types={'TRAIN', 'TEST', 'OBS',
% 'SURVEY', 'EXP', 'PREDICT'}; TEST and PREDICT can be followed by the test
% set number; OBS can be followed by the number of samples requested.
% <filename>.premanipvar, <filename>.manipvar, <filename>.postmanipvar:
% One line with a list of variables.
% <filename>.premanipval, <filename>.manipval, <filename>.postmanipval:
% Multiline matrix, each line corresponding to one instance.
% Each column corresponds to a variable.
% The query object stores this information in corresponding fields.
% Missing values are coded as NaN.
% Methods:
% load --       called by the constructor to load the this.
% save --       save the query to file
% struct --     show the structure
% xml_display -- show structure as XML
% In addition, many fields can be set/get with a method having the same
% name; see methods(query)

% Isabelle Guyon -- isabelle@clopinet.com -- May-Oct 2009
```

```
classdef query
    properties (SetAccess = private)
        % Identification
        participant_ID=[];
        model_name=[];
        experiment_name=[];
        date_submitted=[];
        type='OBS';
        README='';
        % Inputs
        sample=[];
        manipvar=[];
        manipval=[];
        postmanipvar=[];
        % Auxiliary fields and output fields
        postmanipval=[]; % Values for the queried "queryvar" variables
        premanipvar=[]; % List of variables in the premanipval array
        premanipval=[]; % Init. values of system before manipulating it
              % This is used if particular subjects are chosen on which
              % to perform a given manipulation rather than drawing
              % them at random according to the natural distribution.
              % This is also used to return training or test data.
        predict=[];  % Same as postmanipval
        varnum=0; % Total number of var. (not including the target)
                  % i.e. the maximum variable index value
        samplenum=[]; % Number of samples asked for
        manipnum=[]; % Num. of time a var. is manipulated in the query
        obsernum=[]; % Num. of time a var. has been observed (value
                     % requested) in the query, including targets.
        targetnum=[]; % Number of times a target variable is observed
        % Private
        types={'TRAIN', 'TEST', 'OBS', 'SURVEY', 'EXP', 'PREDICT'};
    end
    properties
        % All the costs are computed by the model
        samplecost=0; % samplenum * cost_per_sample
        manipcost=0; % manipnum * cost_per_manip
        obsercost=0; % obsernum * cost_per_observation
        targetcost=0; % targetnum * cost_per_target (this is on top of
                      % the regular cost for observing a variable: the
                      % target may be more expensive
        totalcost=0; % Overall cost
        % Prediction score and estimated error bar
        score=[];
        ebar=[];
        % Flag(s)
        is_overbudget=0; % Budget exceeded
    end
end
```

```
%=========================================================================
% MODEL template for a data generative model
%=========================================================================
% M=model(hyper)
% hyper -- Hyperparameters (public properties). Use the Spider syntax.
%   train_num (Size of the training set)
%   test_num (Size of the test set)
%   cost_per_sample
%   cost_per_target_observation
%   cost_per_var_observation
%   cost_per_var_manipulation
% By default, all values are set to zero.
% If a config file model_config.txt exists, the hyperparameters are
% loaded from that file. Any hyperparameter specified as argument
% overwrites the default or configuration values.
%
% Examples:
%   default(model) % list the default values of the HP
%   M=model;
%   M=model('train_num=3000');
%   M=model({'train_num=3000', 'cost_per_sample=234'});
%   M.task_n_pricing or task_n_pricing(M)
%   B=initial_budget(M);
%   var_profile(M)  % shows the variable properties
%   [Q, M]=process_query(M, query);

% Isabelle Guyon -- isabelle@clopinet.com -- October 2009

classdef model
    % <<------------- hyperparameters ----------------->>
    properties
        train_num=0; % load_config/evel_hyper: Size of the training set
        test_num=0;  % load_config/evel_hyper: Size of the test set(s)

        cost_per_sample=0;              % load_config/evel_hyper
        cost_per_target_observation=0;  % load_config/evel_hyper
        cost_per_var_observation=0;     % load_config/evel_hyper
        cost_per_var_manipulation=0;    % load_config/evel_hyper
    end
    % <<------------- model ----------------->>
    properties (SetAccess = protected)
        time_dept=[];       % init: Time dependency (0/1)
        target=[];          % init: Indices of the target variables
        observable=[];      % init: Indices of the observable variables
        actionable=[];      % init: Indices of the actionable variables
        unobservable=[];    % init: Indices of the unobservable var.
        train_cost=0;       % Price of the default labeled training set
        test_cost=0;        % Price of the default test set
        initial_budget=0;   % Initial budget
    end
end
```

Getting started guide

You may either use the Virtual Lab interface available from
http://www.causality.inf.ethz.ch/workbench.php
Or download the Matlab package GLOP from
http://www.causality.inf.ethz.ch/repository.php?id=23.
Working directly with GLOP will allow you to quickly perform trial experiments on example models. The Virtual Lab interface may then be used to get familiar with the query protocol. The Virtual Lab will be used in benchmarks and challenges to evaluate methods on new unknown models.

Getting started with GLOP

Installation

Unzip glop.zip
Open Matlab and go to the GLOP directory and type **use_glop** at the prompt.

Finding your way around

```
To know the list of models, type
> whoisglop
To know the default values of a model hyperparameters:
> default(model)
To know the methods:
> methods(model)
To know the properties:
> properties(model)
```

Examples

```
> q=query('OBS 20');
or
> q=query('TRAIN');
or
> q=query('TEST');
and
> a=alarm({'cost_per_sample=1', 'cost_per_var_observation=1',
'cost_per_var_manipulation=2'});
or
> a=sprinkler;
etc.
> initial_budget(a)
> task_n_pricing(a)
> [q, a]=process_query(a, q);
```

Getting started with the Virtual Lab

You may follow these simple steps:

- Investigate the models by clicking on the links in the Virtual Lab **Index** page. Choose a model you want to work on.
- Use the format described on the **Info** page to format your query.
- Register by entering your personal information in the **Login** page (or just login if you are already registered).
- Upload your query packaged as a zip archive using the **Upload** page form.
- Retrieve the answer (your data) from the **My Lab** page.

How to design experiments?

Here is a brief outline of the steps taken in experimenting and causal modeling:

1. **Problem specification**: Define your **problem** and your **goals** . In the Virtual Lab, problems are already formalized.
2. **Feature set definition**: Identify potentially **relevant factors** . In the virtual lab the feature set is already given: those are the system variables. In some cases, the task designer may hide a number of variables to test the robustness of algorithms against **hidden confounders**, which are unknown common causes to several variables in your system.
3. **Manipulation protocol**: Figure out how to perform actions on the system and **manipulate variables** of interest. This step is often very complex in real experiments because we do not always have easy means of influencing variables individually as an external agent. Not all variables are actionable or even observable. Some may be unethical to manipulate. In the Virtual Lab, things are simple: we tell you which variables are actionable. All you have to do to carry out experiments is to **initialize or clamp desired variables**.
4. **Experimental design**: Given a budget (here you have "virtual cash"), spend it in data collection, observations, and manipulations to achieve the goals you have set to yourselves.
5. **Modeling**: Carry out the experiments and build models with the data collected. Eventually iterate this process until a satisfactory model is obtained. In the Virtual Lab, all you have to do is to submit queries via the Upload Page using the format described below. Your virtual cash account will be automatically debited and you will be able to download the results of your experiments your private Mylab page.
6. **Deployment**: Deploy your model to predict the consequences of actions in new situations. In the Virtual Lab, we provide you with test data, which was drawn from a post-manipulation distribution. The manipulations are performed by the task designers. depending on the task, the designers may or may not inform you of what exact manipulation(s) was performed in test data. When you are done with modeling and **before your run out of virtual cash**, you must ask for the test data. **WARNING:** The test data will cost you virtual cash, so make sure you keep enough virtual cash. We do not withhold from your cash account a fixed amount to pay for the test data because, if you cleverly design your experiments and your model, you might get it at a discount price by querying only a subset of the variables. **Once you ask for test data, you must return your prediction results** on test data, no query for more data are allowed.

We will organize competitions in the future. In a competition setup, it will not be possible to work several times on the same task. However, for the time being, you are free to experiment multiple times on the same problem and even to run concurrent experiments with different strategies.

Data formats and protocol

The submission of data requests and prediction results is via the Upload page of the Causality Workbench. A submitted query should be a zip file bundling files described below. Use

```
zip query.zip *
```

or

```
tar cvf query.tar *; gzip query.tar
```

to create valid archives. We provide several examples of queries for the LUCAS model:

1. Observations. Request 25 examples of all the variables. No manipulation is performed. observational data only.
2. Experiment 1.. Request 10 values of the target variable. Most covariate values are provided, except a few missing values.
3. Experiment 2.. Not all variables are manipulated. The pre-manipulation values are given by the selection of training samples.
4. Test data. We ask for the test set 2. Here we ask for postmanip variables, but we will not get them because the test data does not include any postmanipulation observations.
5. Default training set. Training data can be purchased unlabeled, this is cheaper. Then the labels may be queried separately.
6. Survey data. Query asking for a subset of the labels of the default training set.
7. Prediction results. Predictions of the target post-manipulation values on test set 2.

If you want to get baseline results without experimenting, it is always possible with the initial budget to buy the **default training set** and the **entire test set**. Just submit two (separate) queries with a **single query file**, each containing a **single word**:

1. To get training data, write the word **TRAIN** on the first line.
2. To get test data, write the word **TEST** on the first line.

File formats for data queries and prediction results

Filename	Non-experimental data	Experimental data	Survey data	Prediction results	Description	File Format
[submission].query	Compulsory (TRAIN, TEST [n] or OBS [num])	Compulsory (EXP)	Optional (SURVEY)	Optional (PREDICT [n])	Type of query.	A single key word on the first line, optionally followed by a number on the same line: TRAIN: get the default training set. TEST [n]; replace [n] by 1, 2, 3 to get the nth test set TEST=TEST 1. OBS [num]: get observational data; replace [num] by the number of samples requested. EXP: get experimental data (the number of samples is determined by

						the number of lines in [submission].sample and [submission].manipval). SURVEY: get training labels. PREDICT [n]: replace [n] by 1, 2, 3 to indicate that predictions correspond to the nth test set. PREDICT=PREDICT 1.
[submission].sample	NA	Optional	Compulsory	NA	Sample ID in the default training set. The corresponding samples are used to set premanipulation values.	A list of sample numbers, one per line (the numbering is 1-based and corresponds to lines in the training data).
[submission].premanipvar	Optional	Optional	Optional	NA	List of the pre-manipulation variables (observed before or without experimentation). By default (no file given): (1) for non-experimental and experimental data: all the observable variables, except the target; (2) for survey data: the target.	A space-delimited list of variable numbers on the first line of the file. All variables are numbered from 1 to the maximum number of visible variables, except the target variable (if any), which is numbered 0.
[submission].manipvar	NA	Compulsory	NA	NA	List of the variables to be manipulated (clamped).	
[submission].postmanipvar	NA	Compulsory	NA	Optional	List of the post-manipulation variables (observed after experimentation). By default: the target variable.	
[submission].premanipval	NA	NA	NA	NA	Not applicable: use [submission].sample to initialize values of the pre-manipulation variables.	Each line corresponds to an instance (sample) and should contain space delimited variable values for all the variables of that instance. Use NaN if the value is missing or omitted. The number of lines in [submission].manipval should match the number of samples in [submission].sample (if provided). You may omit [submission].query and provide [submission].predict instead of [submission].postmanipval if there is a single test set and no experiments are involved.
[submission].manipval	NA	Compulsory	NA	NA	Clamped values for the manipulated variables, listed in [submission].manipvar.	
[submission].postmanipval or [submission].predict	NA	NA	NA	Compulsory	Predictions values for all the samples of TESTn.	

File formats for data received and prediction scores

Data archives with the training or test data you requested are available from your private Mylab a short time after you placed your query. Prediction score are also displayed on the Leaderboard page.

Filename	Non-experimental training data	Survey data	Experimental training data	Test data	Evaluation score	Description	File Format
[submission].query	Optional (TRAIN or OBS)	Optional SURVEY	Optional EXP	Optional TEST	Optional PREDICT	Type of query.	One keyword and optionally a number on the first line (copied from the query submitted).
[answer].premanipvar	Optional	NA	Present if requested	Present	NA	List of the pre-manipulation variables. By default, all the observable variables except the target.	A space-delimited list of variable numbers on the first line of the file. All variables are numbered from 1 to the maximum number of visible variables, except the default target variable (if any), which is numbered 0. If the file is missing or empty, an empty list is assumed.
[answer].manipvar	NA	NA	Present	Optional	NA	List of the variables to be manipulated (clamped).	
[answer].postmanipvar	NA	NA	Optional	Optional	NA	List of the post-manipulation variables. By default: the target variable 0.	
[answer].premanipval or [answer].data	Present	NA	Present if requested	Present	NA	Pre-manipulation values.	Each line corresponds to an instance (sample) and contains space delimited variable values for all the variables of that instance (or a single target value for [answer].label files). [answer].data files contain unlabeled default training data for problems without experimentation
[answer].label	NA (to get the target variable values, use the index 0)	Present	NA (to get the target variable values, use the index 0)	NA	NA	Target values for default trainign examples. Equivalent to [answer].premanipval when [answer].premanipvar (with the single value 0) is omitted.	
[answer].manipval	NA	NA	Present	Optional	NA	Clamped values for the manipulated variables, listed in [answer].manipvar. Those correspond to manipulations performed by the organizers so they are free of charge.	
[answer].postmanipval	NA	NA	Present	Hidden to the participants	NA	Post-manipulation values. In answer to [submission].postmanipvar	
[answer].is_overbudget	Optional	Optional	Optional	Optional	Optional	File indicating that the budget was overspent and the query was not	The value1.

						processed.	
[answer].score	NA	NA	NA	NA	Present	Prediction score.	A numeric value.
[answer].ebar	NA	NA	NA	NA	Present	Error bar.	A numeric value.
[answer].varnum	Present	Present	Present	Present	NA	The total number of observable variables (excluding the target).	A numeric value.
[answer].samplenum	Present	Present	Present	Present	NA	Number of samples requested.	A numeric value.
[answer].obsernum	Present	Present	Present	Present	NA	Number of variable values observed (including the target).	A numeric value.
[answer].manipnum	Present	Present	Present	Present	NA	Number of values manipulated.	A numeric value.
[answer].targetnum	Present	Present	Present	Present	NA	Number of target values observed.	A numeric value.
[answer].samplecost	Present	Present	Present	Present	NA	Cost for the samples requested (labeled samples may cost more than unlabeled samples).	A numeric value.
[answer].obsercost	Present	Present	Present	Present	NA	Cost for the observations made.	A numeric value.
[answer].manipcost	Present	Present	Present	Present	NA	Cost for the manipulations made.	A numeric value.
[answer].targetcost	Present	Present	Present	Present	NA	Additional cost for target observations.	A numeric value.
[answer].totalcost	Present	Present	Present	Present	NA	Total cost.	A numeric value.

In addition, summary information is provided to the central server via an XML format. Both “models” and “queries” have data structures, which can be summarized in XML. We give below typical examples.

1) Following request from the Causality Workbench server to the remove server named “GLOP” to list its models, the remote server returns the following information, which is then formatted as a table in the “Index” page of the Virtual Lab:

```
<virtual_lab name="GLOP" job_id="job985873">
      <version> alpha version - Oct 27 2009 </version>
      <models num="8">
            <model name="alarm">
                  <train_num value="2000" />
                  <test_num value="10000" />
                  <cost_per_sample value="9" />
                  <cost_per_target_observation value="0" />
                  <cost_per_var_observation value="2" />
                  <cost_per_var_manipulation value="4" />
                  <time_dept value="0" />
                  <num_target value="0" />
                  <num_observable value="37" />
                  <num_actionable value="37" />
                  <num_unobservable value="0" />
                  <train_cost value="166000" />
```

```
            <test_cost value="830000" />
            <initial_budget value="996000" />
            <current_budget value="996000" />
        </model>

        <model name="calus">
            ...
        </model>

        [etc. more models here]

        <model name="sprinkler">
            ...
        </model>
    </models>
</virtual_lab>
```

2) Following the same request, the remote server creates individual profile files for the models, for information to the participants (those are linked from the model table in the "Index" page):

```
<model name="lucas">
    <train_num value="2000" />
    <test_num value="10000" />
    <cost_per_sample value="10" />
    <cost_per_target_observation value="40" />
    <cost_per_var_observation value="1" />
    <cost_per_var_manipulation value="10" />
    <time_dept value="0" />
    <num_target value="1" />
    <num_observable value="12" />
    <num_actionable value="9" />
    <num_unobservable value="0" />
    <train_cost value="124000" />
    <test_cost value="210000" />
    <initial_budget value="334000" />
    <task>LUCAS: This is a Bayesian network simulator for a toy
      problem inspired by the problem of identifying risk
      factors of lung cancer. See
      http://www.causality.inf.ethz.ch/data/LUCAS.html , etc
     </task>
     <variables num="12">
        <![CDATA[
        Index   Name    Access  Type    Min     Max
        0       Lung cancer     observable      binary  -1      1
        1       Smoking actionable      binary  0       1
        2       Yellow Fingers  actionable      binary  0       1
        3       Anxiety actionable      binary  0       1
        4       Peer Pressure   actionable      binary  0       1
        5       Genetics        observable      binary  0       1
        6       Attention Disorder      actionable      binary  0       1
        7       Born an Even Day        observable      binary  0       1
        8       Car Accident    actionable      binary  0       1
        9       Fatigue actionable      binary  0       1
        10      Allergy actionable      binary  0       1
        11      Coughing        actionable      binary  0       1
```

```
      ]]>
      <variable index="0" name="Lung cancer"
        access="observable" type="binary" min="-1" max="1" />
      <variable index="1" name="Smoking" access="actionable"
        type="binary" min="0" max="1" />
      <variable index="2" name="Yellow Fingers"
        access="actionable" type="binary" min="0" max="1" />
      <variable index="3" name="Anxiety" access="actionable"
        type="binary" min="0" max="1" />
      <variable index="4" name="Peer Pressure"
        access="actionable" type="binary" min="0" max="1" />
      <variable index="5" name="Genetics" access="observable"
        type="binary" min="0" max="1" />
      <variable index="6" name="Attention Disorder"
        access="actionable" type="binary" min="0" max="1" />
      <variable index="7" name="Born an Even Day"
        access="observable" type="binary" min="0" max="1" />
      <variable index="8" name="Car Accident"
        access="actionable" type="binary" min="0" max="1" />
      <variable index="9" name="Fatigue" access="actionable"
        type="binary" min="0" max="1" />
      <variable index="10" name="Allergy" access="actionable"
        type="binary" min="0" max="1" />
      <variable index="11" name="Coughing" access="actionable"
        type="binary" min="0" max="1" />
    </variables>
  </model>
```

3) Following a data request from the Causality Workbench to answer a query (a zip file), the remote server packages the answer as a zip file and also returns the following information, which will serve to update the experiment table in the "My Lab" page of the Virtual Lab:

```
<virtual_lab name="GLOP" job_id="job756414">
      <experiment pid="isabelle" model="calus" name="experiment1"
date="2009-10-24-180909">
            <model name="calus">
                  <train_num value="5000" />
                  <test_num value="5000" />
                  <cost_per_sample value="3" />
                  <cost_per_target_observation value="0" />
                  <cost_per_var_observation value="2" />
                  <cost_per_var_manipulation value="5" />
                  <time_dept value="0" />
                  <num_target value="0" />
                  <num_observable value="12" />
                  <num_actionable value="8" />
                  <num_unobservable value="0" />
                  <train_cost value="135000" />
                  <test_cost value="135000" />
                  <initial_budget value="270000" />
                  <current_budget value="149325" />
            </model>
            <query type="OBS 25">
                  <premanipvar dim1="1" dim2="12" />
```

```
                  <premanipval dim1="25" dim2="12" />
                  <varnum value="0" />
                  <samplenum value="25" />
                  <manipnum value="0" />
                  <obsernum value="300" />
                  <targetnum value="25" />
                  <samplecost value="75" />
                  <manipcost value="0" />
                  <obsercost value="600" />
                  <targetcost value="0" />
                  <totalcost value="675" />
                  <is_overbudget value="0" />
            </query>
      </experiment>
</virtual_lab>
```

3) Following a scoring request of prediction results from the Causality Workbench (a PREDICT query formatted as a zip file also), the remote server packages the answer as a zip file and also returns the following information, which will serve to update the "Learderboard" page:

```
<virtual_lab name="GLOP" job_id="job987618">
      <experiment pid="isabelle" model="sprinkler" name="experiment2"
date="2009-10-26-234155">
            <model name="sprinkler">
                  <train_num value="2000" />
                  <test_num value="10000" />
                  <cost_per_sample value="8" />
                  <cost_per_target_observation value="0" />
                  <cost_per_var_observation value="1" />
                  <cost_per_var_manipulation value="2" />
                  <time_dept value="0" />
                  <num_target value="0" />
                  <num_observable value="4" />
                  <num_actionable value="2" />
                  <num_unobservable value="0" />
                  <train_cost value="24000" />
                  <test_cost value="120000" />
                  <initial_budget value="144000" />
                  <current_budget value="24000" />
            </model>
            <query type="PREDICT">
                  <varnum value="0" />
                  <samplenum value="0" />
                  <manipnum value="0" />
                  <obsernum value="0" />
                  <targetnum value="0" />
                  <samplecost value="0" />
                  <manipcost value="0" />
                  <obsercost value="0" />
                  <targetcost value="0" />
                  <totalcost value="0" />
                  <score value="0.492329" />
                  <is_overbudget value="0" />
            </query>
      </experiment>
</virtual_lab>
```