

ELTOY: Implementing Bayesian Computation Through Constraint Propagation

Russell G. Almond¹

University of Washington, Department of Statistics

ABSTRACT

ELTOY is a program for teaching Bayesian statistics and eliciting prior distributions through interactive graphics. The posterior distribution is constrained to the prior distribution and the data so that changes to one are automatically propagated to the others. Because the constraints are associated with conjugate family objects, ELTOY is easily extendable. This paper describes the program ELTOY, the object-oriented and constraint propagation techniques used in the design, and how to obtain a copy of ELTOY (which is free).

¹ Russell Almond is currently at Statistical Science, Inc. (StatSci), 1700 Westlake Ave, N., Suite 500, Seattle, WA 98109, almond@statsci.com. The bulk of this work was done while the author was at the University of Washington.

1.0 Overview of ELTOY

Good(1976) compares Bayes theorem to a black box. You (the term *You* in this paper refers to an abstract decision maker, loosely in the spirit of Savage and de Finetti) input *data*—observations about a phenomenon of interest,—a *prior* model—information, independent from the data, about some *parameter* useful for describing the phenomenon,—and a *likelihood* model—information which describes the relationship between the *parameter* and the *data*. Bayes Theorem is a rule which allows the data, prior and likelihood to be combined to produce posterior inferences in the form of a probability distribution about the parameter. Figure 1 illustrates this simple concept.

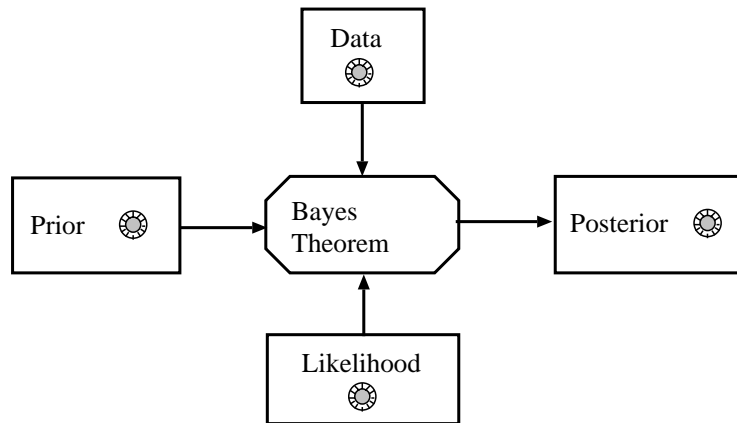


Figure 1. Bayes

Good goes on to state that by wiggling the inputs to the black box, You can study the sensitivity of the output to the inputs. The “knobs” on the boxes in Figure 1 are meant to suggest this idea. Wiggling the prior assess Bayesian robustness, wiggling the likelihood assess classical robustness, and wiggling the data assess classical resistance. Good uses the black box as a metaphor for these sensitivity analyses, but recent advances in computing technology make it possible for this metaphor to implemented and used as a tool for teaching Bayesian statistics.

ELTOY (Elicitation Tool for You) is a computer program which illustrates the Bayesian

paradigm. It runs in XLISP-STAT (Tierney, 1990) which provides a platform for experimenting with user interfaces for statistics. `ELTOY` displays graphs of the prior and posterior distribution of the parameters and attaches controls (sliders instead of knobs) to the hyperparameters of these distributions and to the data. Thus You (the term *You* also refers to the user of `ELTOY`) can change the hyperparameters or data and directly observe the effect on the posterior. Thus `ELTOY` is a tool for robust Bayesian analysis.

The design of `ELTOY` is rather different from conventional procedural programs. Bayes theorem is a *constraint* on the posterior distribution—a rule for calculating the form of the posterior distribution given the prior, likelihood and data. Note that this rule is implied by the family of the prior distribution and the likelihood, as is stoted with an *object* representing the conjugate family. `ELTOY` uses an update mechanism, described in section 5, to propgate changes in the prior, likelihood or data to the posterior, thus implementing the functionality implicit in Figure 1.

As the system is currently limited to conjugate prior–likelihood families, `ELTOY` is an advance not in statistical methodology, but rather in the way the information is presented. Consequently, even in this primitive version, `ELTOY` is a useful teaching tool. Given that the process of eliciting knowledge in the form of a probability distribution from a domain expert often consists of teaching that expert the meaning of the chosen prior distribution, `ELTOY` may yet prove useful in elicitation. Finally, experience with `ELTOY` does indicate where the difficulties will lie in building a more general Bayesian statistics package.

`ELTOY` consists of three different tools: A distribution display tool—`Dist-toy`, a central limit theorem illustration—`CLT-TOY`, and the conjugate Bayesian display tool—`ELTOY`. Figure 2 shows the main menu bar from which You can select both the tool. Pull down menus from each item in the menu bar allow You to select from a library of distribution family objects. Sections 2–4 of this paper describe these three tools. Section 5 discusses some of the issues raised by the implementation of

ELTOY and Section 6 the problems inherent in scaling the program up to handle larger and less constrained models.



Figure 2. menubar

As ELTOY is an interactive computer program, this paper cannot capturing its true nature. This paper can merely describe how to operate ELTOY and some of the design decision which went into its construction. As this paper is not a complete description of this research, interested readers are encourage to get the software. It is free and it is available through anonymous ftp from `ftp.stat.washington.edu` and via email from `statlib@stat.cum.edu`.² XLISP-STAT, in which ELTOY runs, is also free; XLISP-STAT is available through `statlib` or via anonymous ftp from `umnstat.umn.edu`. Installation instructions and the user guide are distributed with the source code.

2.0 Dist-toy—Displaying probability distributions.

An important subtask in building ELTOY was simply building a tool for dynamically displaying a probability distribution. As this is an interesting teaching tool in its own right, it was left as a separate piece of ELTOY called `dist-toy`.

`Dist-toy` shows two views of a parametric probability distribution. The first is a control box which shows the parameters and allows the user to manipulate them. The second is a graph which shows the shape of the distribution, a p.d.f. for continuous distribution families and a p.m.f. for

² Statlib is an automatic email server maintained by Mike Meyers at Carnegie-Mellon University. For instruction send electronic mail with the single line “send index” (no subject is required) to the `statlib` address.

discrete families. These two views are linked so that changes in the parameters are immediately reflected in the graph.

ELTOY represents probability distributions with XLISP-STAT's object system (Tierney, 1990). XLISP-STAT uses a prototype based system so each probability distribution family is both itself an object and an object prototype from which instances (probability distributions of that type) can inherit values and methods. For example, the prototype for all probability distributions contains a default method for finding the median (taking the .5 quantile). However, the normal family prototype overrides that method with the simple method of returning the mean.

We can also see the effect of this object oriented design in the way ELTOY chooses between the p.d.f. and p.m.f. ELTOY defines two sub-classes of a probability distribution: discrete and continuous. The graphical display method for discrete distributions displays the probability mass function (p.m.f) and the method for continuous distributions displays the probability density function (p.d.f.). Thus, the Poisson and Binomial families inherit from the discrete distribution prototype and hence display the p.m.f., and the Normal and Gamma inherit from the continuous distribution prototype and hence display the p.d.f.

Much of the information in the distribution family relates to the parameters of the distribution (these become the hyperparameters in the ELTOY component of the program). In particular, a distribution family defines the limits of each parameter value, whether or not the parameter is restricted to integer values, and more complex constraints among several parameters can be represented by a constraint function (For example, in the uniform distribution over the interval $[a, b]$ it must be true that $a < b$. ELTOY represents this constraint with the Lisp function `#'(lambda (a b) (< a b))`, which will return "true" if and only if $a < b$. Any candidate set of parameters are passed to this function and if the function does not return true the system reports an error to the user.) All this complex information on parameters is used to implement the "knobs" on the

parameters (hyperparameters) of the distribution. ELTOY can use the knowledge about parameters of the probability distribution family object to set up sensible limits and granularity for the sliders and to prohibit the user from inputting mathematically nonsensical values.

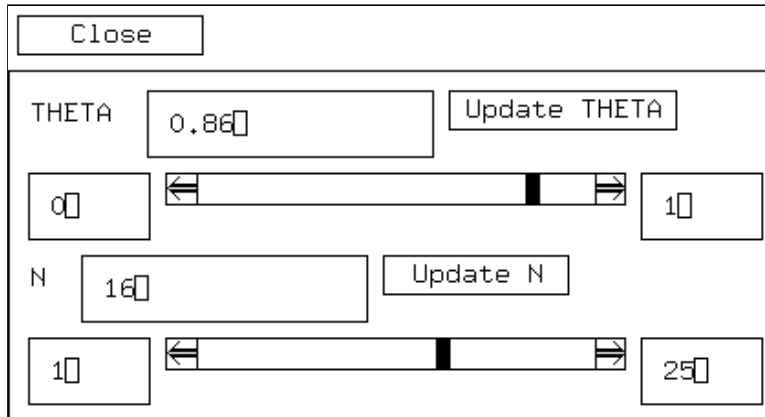


Figure 3. bislider

The parameter controls (Figure 3) are simple to operate. To change the value of one of the parameters You move the indicator with the mouse. The plot of the distribution (Figure 4) automatically changes to reflect the new values of the parameters. For finer control, You can edit the value of the parameter directly, or extend the range of the slider by editing the endpoints. The buttons marked “Update *parameter*” (Figure 3) are used to register the new values. (A limitation of LISP-STAT is that it will not accept new values on carriage return or some similar keystroke event.)

The graph does not automatically rescale itself when the data points move out of range. This causes the distribution to distressingly wander off the page. This is a deliberate design decision. Think what would happen to the normal distribution as You change the mean and variance if the scales automatically changed; only the labels on the axis would change (this makes a nice classroom demo). Pressing the menu button on the graph (or selecting the “Graph” menu on the MacIntosh version) reveals a list of options, one of which allows You to manually rescale the graph.

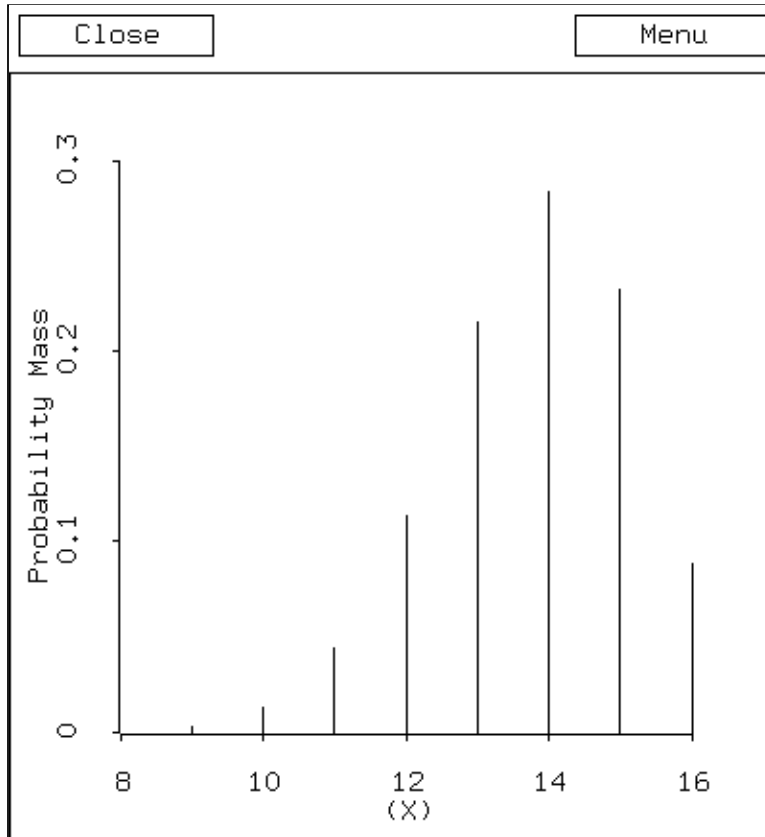


Figure 4. bipmf

The difficulty with automatic scaling in `ELTOY` reflects a common problem in the design of dynamic statistical graphics: automatically changing the scaling to capture maximal information may cause the the user to loose context information—the scale of the graph. There is no simple resolution to the conflicting goals of keeping the data on the screen and preserving the context of the scaling information. The optimal design might be to make that option settable by the user (e.g., by setting a global variable, or checking a box on an options list). Better still would be to provide universal zooming and scaling interfaces to all scatterplots, and put these decisions under direct user control.

`Dist-toy` is implemented as a simple series of constraints. The value above the slider is constrained to equal the parameter value and the position of the slider is constrained to be

the value of the parameter relative to the endpoints of the slider. Similarly, the shape of the picture in the graph is constrained by the values of the parameter. The constraints are maintained hierarchically so that a request to change a parameter is passed to the highest object in the hierarchy (in this case an invisible Dist-Tool object), that object then passes an `:update` message down to all of its components (in this case, the slider and the graph).

This hierarchical constraint propagation scheme is simple to extend. For example, one could imagine placing an active point on the graph corresponding to the mode of the p.d.f. (or with a little more imagination, the p.m.f.). Dragging this point would generate a request to change the mode of the distribution. This request would be passed up to the highest level object which could then turn it around as a update message to the parameters. Both Section 5.0 and documentation supplied with `ELTOY` describe this hierarchical updating scheme in more detail.

Because `dist-toy` relies on the LISP-STAT object system, virtually the same mechanism will work for any parametric distribution family. In fact, the `dist-toy` library contains most of the familiar members of the exponential family. The system is easy to extend; any new subclass of the existing probability distribution classes which follows the established protocol (described in the `ELTOY` documentation) will work in the place of the binomial distribution shown in the example.

3.0 CLT-TOY—Central Limit Theorem Demonstration

One advantage of object oriented programming is that it promotes code reuse; `CLT-toy` is an example. The implementation of `dist-toy` included a library of standard parametric distribution families. `CLT-toy` reuses that library to build a program to demonstrate the central limit theorem. Adding methods for drawing numbers to the protocol for probability distributions enables `CLT-toy` to use the same probability distribution family objects in this new context.

Figure 5 shows an example histogram for 500 samples, each consisting of the sum of two draws from a “Holey” distribution (an artificial distribution designed to have slow convergence

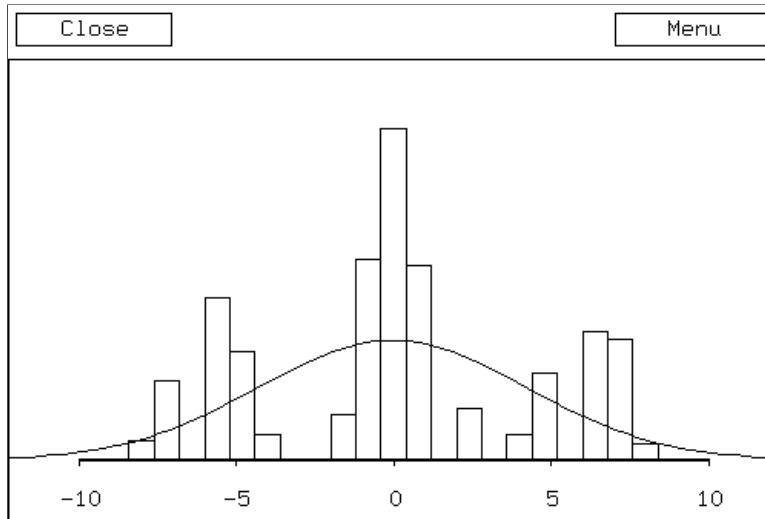


Figure 5. clthist

to the central limit theorem). The controls for the CLT-TOY (shown in Figure 6) allow You to draw another 500 samples and add them to the existing samples to make a sample of the sum of 3 draws, and so forth. The normal curve superimposed on this histogram (Figure 5) provides a visual indication of convergence.

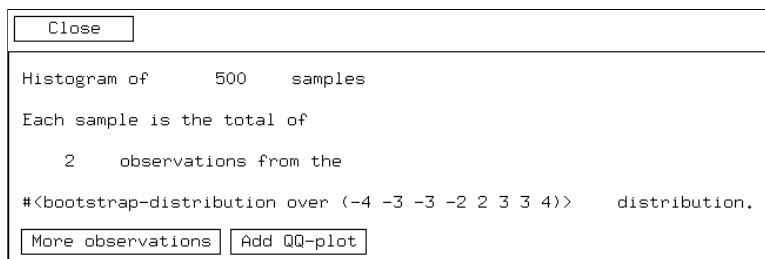


Figure 6. cltcontrols

A better way to watch the convergence to normality is to look at a Quantile-Quantile plot of the sample distribution versus the normal distribution. A button on the control panel allows You to add this plot; Figure 7 shows the result. The “Holey” distribution shows a very interesting pattern,

which straightens out as more samples are added. This toy is a good classroom aid for teaching not only the central limit theorem, but for illustrating what influences its rate of convergence.

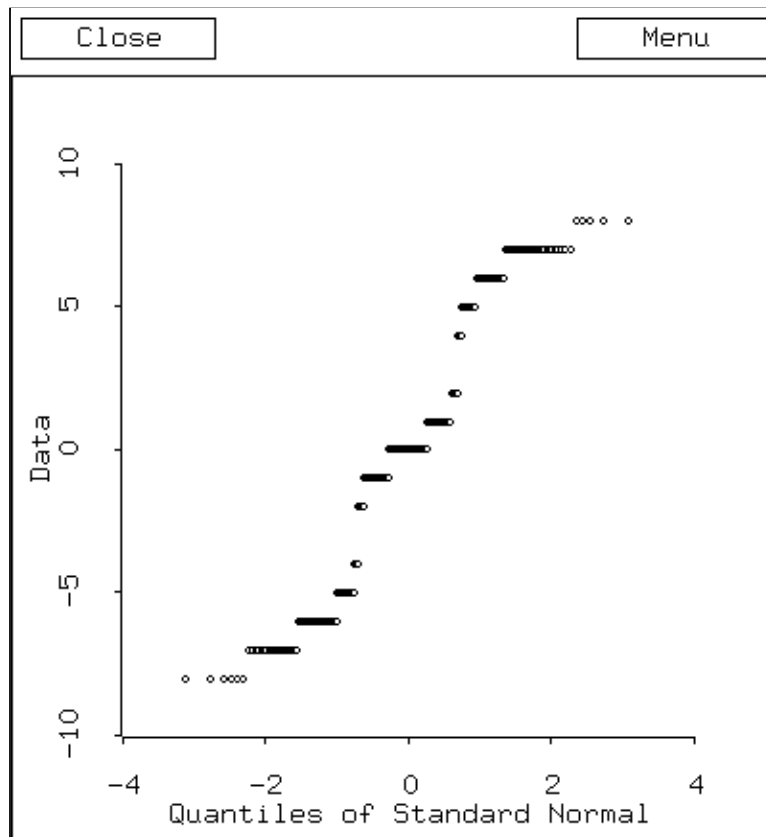


Figure 7. cltqq

In order to facilitate entering unusual distributions (such as “Holey”), the distribution object protocol for CLT-toy was extended to include two non-parametric families of distributions. One, simply called the `finite-family`, which allows an arbitrary number of atoms and an arbitrary probability for each. The second is the `bootstrap-family`, which simply resamples from a sample (a list of values); thus, the probability of any value is proportional to the number of times its appears in the list.

4.0 ELTOY—Full Bayesian models

The program ELTOY was designed to allow the elicitation of prior distributions. It does this by displaying two Dist-Tool (the heart of `dist-toy`) objects, one for the prior and one for the posterior. By linking these displays, changes in the prior are reflected in the posterior and visa-versa. Figure 8 shows the controls for the prior and posterior distributions and Figure 9 shows the actual prior and posterior displays. In this example, the family is beta-binomial, so both the prior and posterior displays show beta distributions.

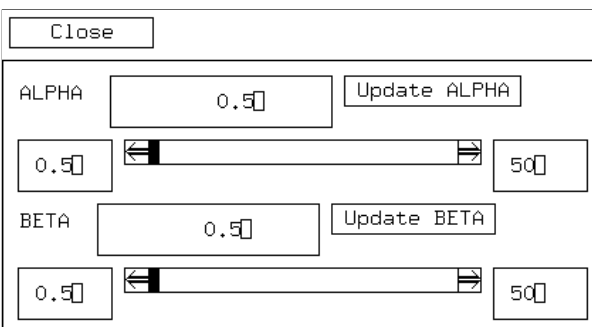


Figure 8a. ppslider

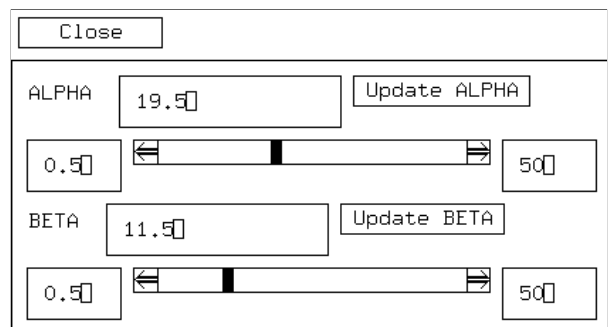


Figure 8b. Prior Controls

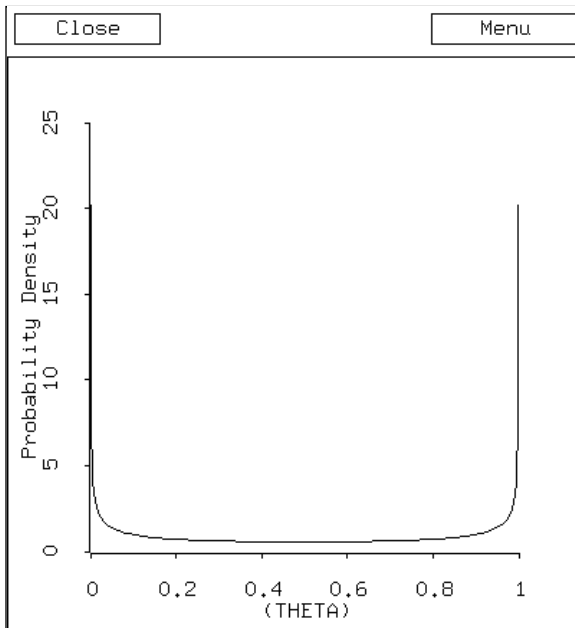


Figure 9a. pppdf

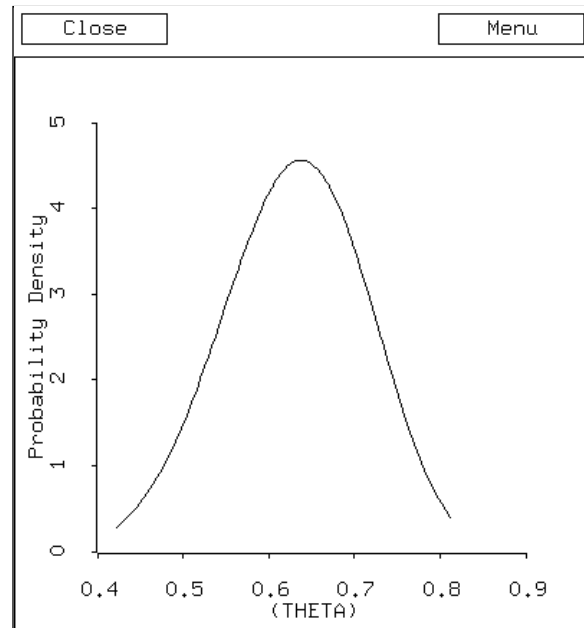


Figure 9b. Prior p.d.f.

The prior and posterior distributions are linked through a *Data-Link* object which links a posterior distribution to a prior distribution and a set of data (and nuisance parameters). Figure 10 shows the controls for binomial data. In this case, the data are a series of binomial observations showing 8, 6 and 5 events respectively out of 10 trials each time. The data is entered as a Lisp expression (thus You have access to LISP-STAT variables), and any nuisance parameters (in this case there is just one, the number of Bernoulli trials for each data point) are entered through a slider. (This arrangement of controls is more natural for the normal distribution in which the nuisance parameter is the observation variance and the data is a list of normal observations.)

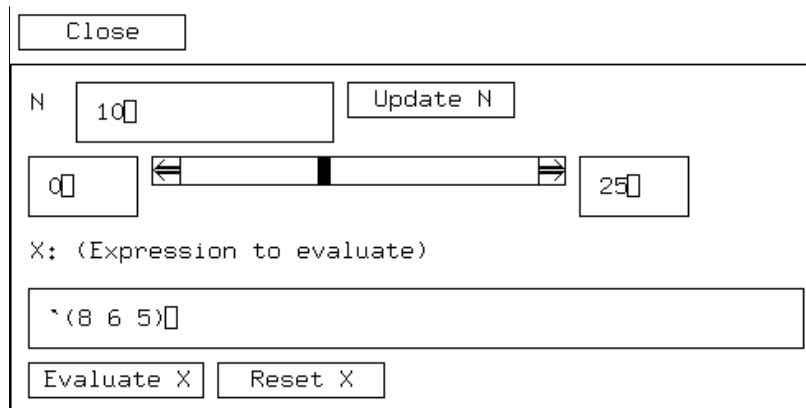


Figure 10. *dataslider*

The update rules work as follows: (1) If You change a prior distribution parameter, the Data-Link computes new posterior parameters and updates the posterior display. (2) If You change the data or a nuisance parameter, the Data-Link computes new posterior parameters and updates the appropriate display. (3) If You change a posterior parameter, the Data-Link computes new prior parameters (if it can, if it can't find a prior associated with that posterior it signals an error) and updates the prior display. This last allows for a type of pre-posterior or "preposterous" sensitivity analysis. This is useful as a learning tool when the data are hypothetical and allows You to develop a prior distribution which has the desired responsiveness to new data. The Data-Link Display

(shown in Figure 11) helps You keep track of what the Data-Link is doing in response to Your actions.

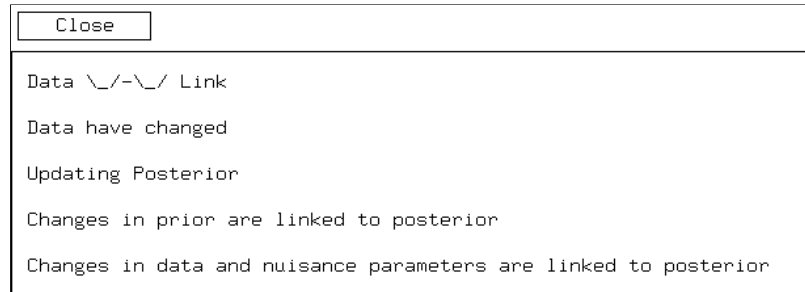


Figure 11. datalink

The Data-Link performs the forward and backwards translation of messages between the prior and posterior display tools. The Conjugate-Family is an object in `ELTOY`, just the way the simple probability distribution is. From the point of view of `ELTOY`, the important thing about Conjugate-Families is that they have methods for the two messages `:forward-link` (calculate posterior parameters from prior and data) and `:reverse-link` (calculate prior parameters from posterior and data). The reverse link is not particularly important if You are willing to sacrifice preposterous analysis. Thus, any method at all could be used to derive the posterior from the prior, even numeric information. It may be necessary to precompute posteriors and interpolate between them in order to achieve the necessary speed for the dynamic graphics, but the `ELTOY` framework can be easily extended to cover a generalized notion of conjugacy.

5.0 The update algorithm

When You change a parameter through an action (such a moving a slider or typing in a new value), `ELTOY` must propagate that change throughout the rest of the system. It uses a hierarchical update algorithm to accomplish that task. As this algorithm could be useful in a number of other task, this section describes it briefly along with some alternative approaches to value propagation.

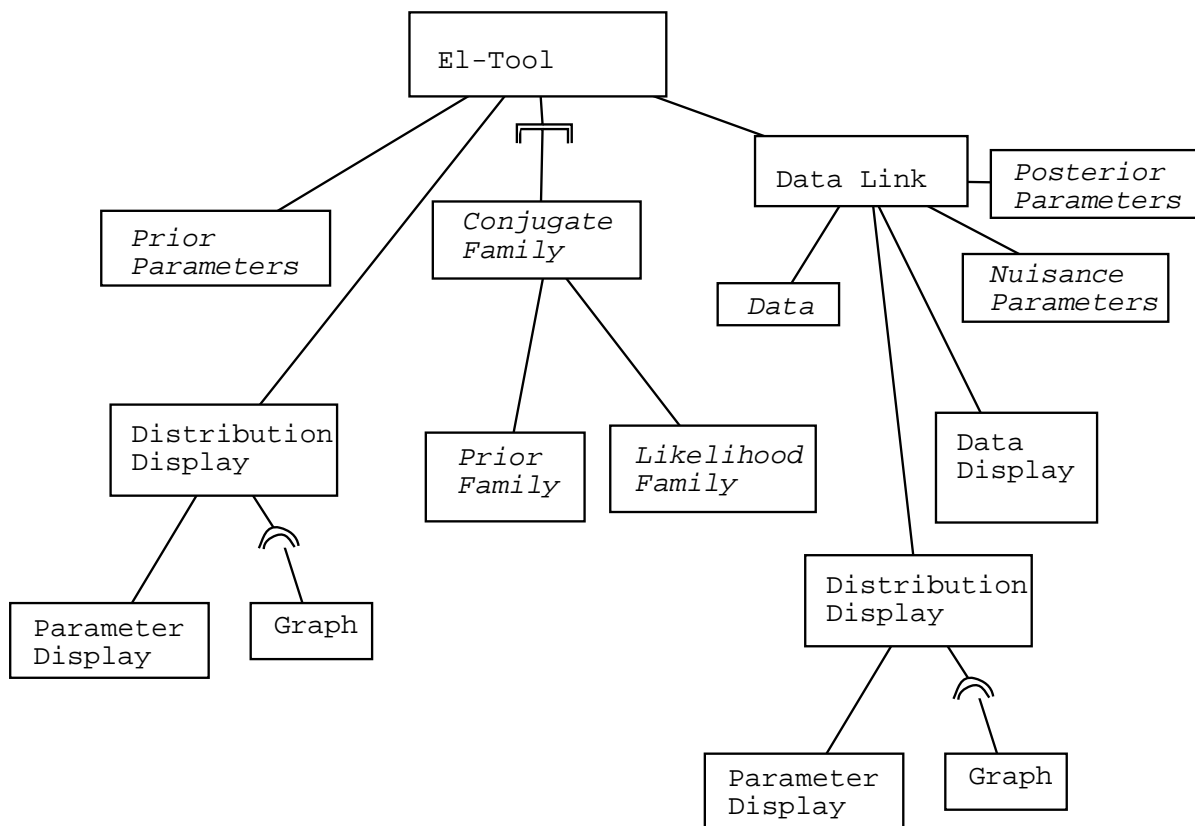


Figure 12. El-Toy

Figure 12 shows the components which make up a typical `ELTOY` invocation. The bottom layers (especially the controls and the graphs) are visible on the screen; the others, higher in the tree, represent invisible components of the system. Objects which are lower in the tree reference information (mostly the parameter values but also the choice of distribution family) belonging to objects higher in the tree. We will call the object immediately above a given object in the `ELTOY` schematic, the *container* of that object, and any object immediately below a *part*. Thus the `El-Tool` object contains every other object in this `ELTOY` session. Note that this “part-of” inheritance is different from the “is-a” inheritance of an object from its prototype or class. Myers, Giuse and

Vander Zanden (1992) describe Garnet “gadgets”, a more formal implementation of this idea.

Suppose You change the value of a prior parameter, for example, You use the slider to set ALPHA to 3.5. The system internally translates this action into a parameters message: `:parameters :alpha 3.5`. The method for handling the `:parameters` message checks to see if the parameters are owned by the object receiving the message (the parameters display for the prior distribution). This is not the case, so it sends the message to its container, the distribution display, which in turn sends its message to the El-Tool, translating it into a `:prior-parameters` message. The El-Tool does indeed own the parameters, so it turns the message into an update message: `:update :prior-parameters :alpha 3.5`.

The update method is the same for every object in the system. The arguments of the update message will always be a another type of message. The first thing the update method does is to resend that message to itself, but this time with a local flag. Thus, the El-Tool object on receiving the `:update` message, sends itself the message `:prior-parameters :local :alpha 3.5`. The method for this message actually sets the parameters. The El-Tool object then sends the `:update` message to each of its its parts. When the distribution display object receives the `:update` message it sends itself a `:prior-parameter :local` message (which it ignores) and sends an `:update` message to both the parameter display and the graph. These update messages force recalculation and display.

The Data-Link object to translates the `:prior-parameter`, `:posterior-parameter`, `:nuisance-parameter` and `:data` messages, passing them to the rest of the distributions. In particular, its local method for the `:prior-parameters` message recomputes the posterior parameters.

Although this message passing scheme is elaborate, it is easy to generalize. Fortunately most

of the work is done by *mixin*³ prototypes which all the update functionality to be easily added to other objects. In particular, most of the functionality associated with referencing parameters from a container object is implemented in Inherited-Parameter-Mixin prototype.

For example, one possible improvement to the system would be to add a mouse mode to the display which allows the user to set the mode of the distribution to a given value (the tools to perform this addition are available in LISP-STAT; it may be added in a future release of ELTOY). The graph would then translate this into an appropriate message which would be subject to the same update algorithm. The update algorithm insures that everybody who needs to know about the changes is informed of them.

The principle alternative to creating an update scheme such as the hierarchical update algorithm used in ELTOY is to employ a constraint maintenance system such as the one employed by Garnet (Myers *et al.*, 1990, Myers, 1992). Garnet uses a system of *formulas*, so that the value of a *slot* in an object (for example the value of a prior or posterior parameter) can depend on other slots in other objects. It uses lazy evaluation, so when the value of the slot is accessed, the object uses the formula to recompute the value of the slot if its value is out of date. Because of this lazy evaluation, it would still be necessary to send an update message to every visible object affected by the change, however, this time it would be sufficient to send the message `:update`, there would be no need for the `:update` message to carry information about the change as happens in ELTOY. Also it would not perform unnecessary calculations, thus it would not update a window which was currently invisible (turned into an icon or behind another window) until that window became visible again.

Although Garnet is capable of handling simple circularity of constraints, it breaks down in the presence of more elaborate dependencies, especially of the kind presented by the Data-Link

³ A *mixin* is a lightweight object which is never intended to be directly instantiated, but merely to have its properties inherited by objects which will actually be used in the system.

object or multiple Data-Links (a possibility supported by `ELTOY`). Multi-Garnet (Sannella and Borning, 1992) uses a more elaborate constraint satisfaction algorithm to handle more complex kinds of constraints. It, however, uses an eager evaluation strategy and so has some of the same disadvantages of the hierarchical update scheme used in `ELTOY`.

Another possible implementation strategy (Sullivan and Notkin, 1990) would use specific mediator objects to register the objects who were interested in a specific change and notify them when changes occur. In the implementation of McDonald and Niehaus(1991), the intermediaries are called “announcements” and the object making a change announces the event. This announcement causes the announcement object to search for the audience of that announcement for that announcer and sends a previously registered notification message to that object. Although the hierarchical update scheme appears to work well with the very hierarchical Bayesian model of `ELTOY`, the use of mediators may work better in less structured problems.

Note that Data-line object is actually a mediator; it translates messages from one part of the system to another. Thus it is the Data-Link object which is responsible for translating messages indicating a change in the prior or the data into a message describing how to change the posterior. It mediates between the announcements made by the prior and those made by the posterior.

6.0 Future Directions

`ELTOY` in its current state has already proven useful as a teaching tool. Although the limitation to conjugate families is severe, by using the generalized notion of conjugacy (an update algorithm rather than a simple formula) that limitation can be easily eliminated. Even the simple case of conjugate families could be useful in certain types of applications, for example eliciting distributions for use in a discrete graphical model (Almond, 1992, Almond, 1993).

It would be nice to include other displays and interaction modes as well. Using the hierarchical update algorithm, it should be possible to extend the functionality to add such things as: alternative

parameterizations of the priors, the ability to move the mean or mode of the distribution with the mouse, or adding predictive distributions (Chaloner and Duncan (1983)). It is already possible to spawn more than one posterior (and data set) for a given prior distribution. (These improvements may come with future releases of `ELTOY` although no releases are planned for the near future.)

Two different generalizations of the system still present substantial technical difficulties in presenting ways for visualizing and manipulating the underlying distributions. Until we can address these difficulties, `ELTOY` will be limited to parametric families and univariate priors.

Expanding `ELTOY` to accept non-parametric families of distributions requires a method for specifying arbitrary distributions graphically. Graphically entering a function is a difficult task. A graphical distribution editor would require mechanisms for expressing constraints such as normalization, symmetry and smoothness without intruding on the Your freedom to specify a distribution. Also it is difficult to express behavior in the tails of the distribution with a picture which clips those tails off.

Expanding `ELTOY` to accept multivariate priors requires a mechanism for visualizing multivariate priors. Simply looking at the marginal distribution for each parameter ignores the possibility of dependence between the parameters. This can appear in posteriors even if it is absent in the priors (Spiegelhalter and Cowell, 1992). Specifying the prior presents another problem. Even in the simple case of the multivariate normal, entering both the mean vector and the covariance matrix is a big task, larger than the bandwidth of current interaction mechanisms like sliders.

One last feature of `ELTOY`: it's fun to use! This, along with many other aspects of such an interactive system are difficult to express in a static medium like a piece of paper. To get the full impact of the ideas presented in this paper, You will need to go get a copy of `ELTOY` and try it out.

References

- Almond, Russell G.(1992).** “GRAPHICAL-BELIEF: Project Overview and Review” Statistical Science Research Report 14. StatSci, 1700 Westlake Ave, N., Suite 500, Seattle, WA 98109. .
- Almond, Russell G. (1993).** *Graphical Belief Models: Algorithms and an Example*. Monograph based on Ph.D. dissertation (Harvard University, Department of Statistics). To be published as a monograph from Van Nostrand Reinhold.
- Chaloner, Kathryn M. and Duncan, George T.(1983).** “Assessment of a Beta Prior Distribution: PM Elicitation.” *The Statistician*. **32**, 174–180.
- Good, I. J. (1976).**, “The Bayesian Influence, or How to Sweep Subjectivism Under the Carpet,” *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science*, Proc. of a Conference in May, 1973, Hooker, C. A. and Harper, W., eds., Vol. 2 (Dordrecht, Holland, D. Reidel, 1976), pp 125-174. Reprinted in *Good Thinking*, University of Minnesota Press, 1983, pp 22–55.
- McDonald, John A. and Niehaus, Mark (1991).** “Announcements: an implementation of implicit invocation.” Technical Report , Dept. of Statistics, U. of Washington, October 1991.
- Myers, Brad A., Guise, Dario A., Dannenberg, Roger B., Zanden, Brad Vander, Kosbie, David S., Pevin Ed, Mickish, Andrew, and Marchal, Phillipe (1990).** “Comprehensive Support for Graphical, Highly-Interactive User Interfaces: The Garnet User Interface Development Environment.” *IEEE Computer* **23**(11), pp 71-85.
- Myers, Brad A. (ed.)(1992).** “The Second Garnet Compendium: Collected Papers 1990-1992.” Technical report CMU-CS-93-108, School of Computer Science, Carnegie Mellon University.
- Myers, Brad A., Giuse, Dario A., Vander Zanden, Brad (1992).** “Declarative Programming in a Prototype-Instance system: Object-Oriented Programming without Writing Methods.” in

Proceedings OOPSLA '92: ACM Conference on Object-Oriented Programming Systems, Languages, and Applications, SIGPLAN Notices, (27) 10, 184–200.

Sannella, Michael and Borning, Alan(1992). “Multi-Garnet: Integrating Multi-Way Constraints with Garnet.” Technical Report 92-07-01, Department of Computer Science and Engineering, University of Washington.

Spiegelhalter and Cowell(1992). “Learning in probabilistic expert systems,” in Bernardo, J.M., Berger, J.O., Dawid, A.P, and Smith, A.F.M., eds., *Bayesian Statistics 4*, Oxford University Press.

Sullivan, Kevin J. and Notkin, David (1990). “Reconciling environment integration and component independence.” In *SIGSOFT'90: Fourth Symposium on Software Development Environments*, Irvine, CA, pp 208–225.

Tierney, Luke (1990). *LISP-STAT: An Object Oriented Environment for Statistical COmputing and Dynamic Graphics*. John Wiley and Sons.

Figure Captions:

Figure 1. Black Box model for Bayesian Computation.

Figure 2. ELTOY Menu Bar (Unix Version).

Figure 3. Slider for Binomial Family Parameters.

Figure 4. Distribution display (p.m.f.) for Binomial

Figure 5. Histogram after 2 draws from “Holey” distribution

Figure 6. Controls for CLT-TOY

Figure 7. Quantile Quantile plot (vs Normal)

Figure 8a. Prior Controls Figure 8b. Posterior Controls

Figure 9a. Prior p.d.f Figure 9b. Posterior p.d.f

Figure 10. Data Controls

Figure 11. Data Link Display

Figure 12. Schematic Diagram of ELTOY components