Proximal/Distal Modeling Framework

C. Bracis

March 27, 2014

1 Fitting model data and generating results

This package "Proximal/Distal Modeling Framework" (pdmod), provides methods to compute the model (Table 1) for different parameter values and fit parameters to experimental data. In this model, Pavlovian conditioning phenomena (acquisition, extinction, spontaneous recovery and the partial reinforcement extinction effect) emerge from reward predictions of parallel neural circuits that combine according to their time-varying uncertainties.

The package is designed for estimating model coefficients from Pavlovian conditioning experiments consisting of trials in which a signal is paired with a reward or non-reward and the response rate of the subject is recorded. Information from experiments consists of the trial time, i.e. the time of the signal, a Boolean indicator of whether a reward is paired with the signal and response rate in each trial. Trials are grouped into sessions or blocks of trials, which are typically separated by larger time intervals than the inter-trial time. An experiment may consist of a single session or multiple sessions.

1.1 Defining trials

The first requirement is to define the experimental setup in terms of the trials performed, meaning the time schedule of the trials, as well as the reward/non-reward outcome of each trial. The class `TimedVector` is used to store this information. For example, suppose the experiment consisted of an acquistion session followed by 4 extinction sessions. The acquisition session was 20, each a minute apart. The extinction sessions started the next day and consited of 5 trials per session with sessions every other day. In the code below, the trials are specified in terms of elapsed minutes and `TV_DAY` is a constant allowing days to easily be added.

```r
library(pdmod)
rewards = c(rep(1, 20), rep(0, 10))
schedule = c(1:10, (2 * TV_DAY):(2 * TV_DAY + 9), (4 * TV_DAY):(4 * TV_DAY +
    1), (6 * TV_DAY):(6 * TV_DAY + 1), (8 * TV_DAY):(8 * TV_DAY + 1), (10 *
    TV_DAY):(10 * TV_DAY + 1), (12 * TV_DAY):(12 * TV_DAY + 1))
```

Table 1: Model equations. Distal ($j = 1$) and proximal ($j = 2$) estimators for trial i, x_i is reward/no-reward outcome $\epsilon(0, 1)$ and Δt_i is the time between trial i and trial $i - 1$.

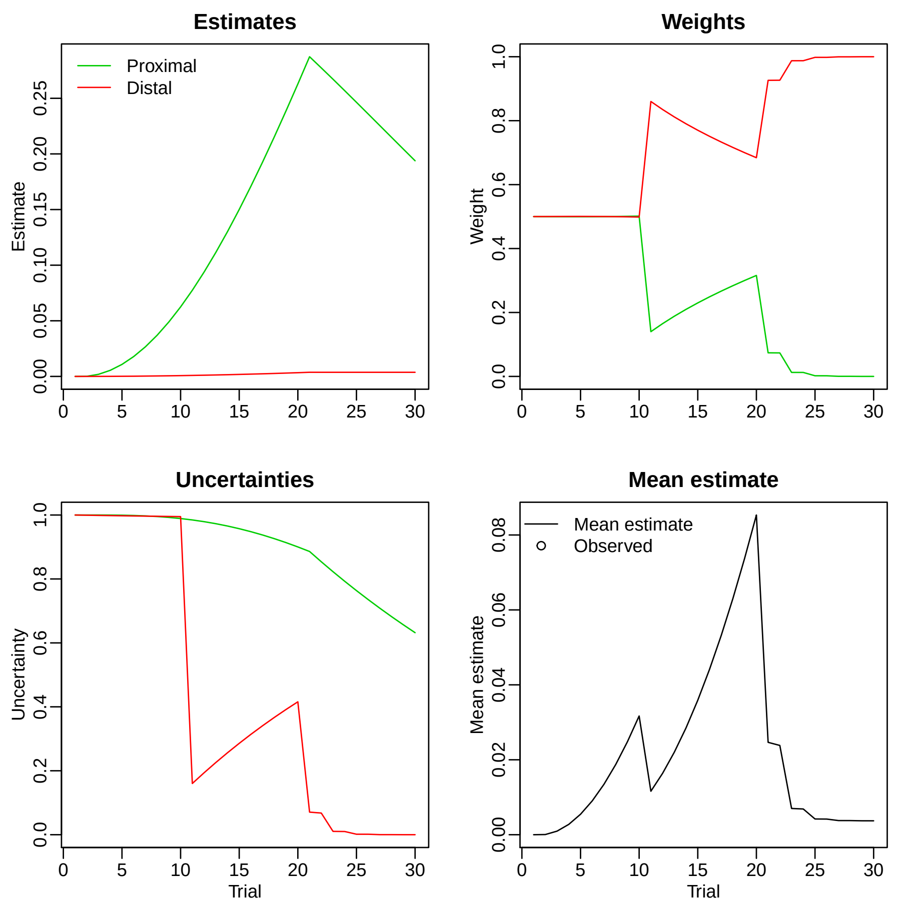
Eq.	Description	Equation	Parameters
1	Reward prediction error j	$\delta_{j,i} = x_i - \hat{x}_{j,i}$	
2	Reward prediction j	$\hat{x}_{j,i} = m_j y_{i-1} \delta_{j,i-1} + \hat{x}_{j,i-1}$	m_j = learning rate $0 < m_1 < m_2 < 1$
3	Uncertainty j unadjusted for Δt_i	$\hat{u}_{j,i} = n(\delta^2_{j,i-1} - \tilde{u}_{j,i-1}) + \tilde{u}_{j,i-1}$	n = learning rate $0 < n < 1$
4	Uncertainty j adjusted for Δt_i	$\tilde{u}_{j,i} = \hat{u}_{j,i} h_j^{\Delta t_i}$	h_j = decay rate $0 < h_1 < 1$, $h_2 = 1$
5	Weight of prediction j	$w_{j,i} = (1/\tilde{u}_{j,i}) \,/\, (1/\tilde{u}_{1,i} + \tilde{u}_{2,i})$	
6	Mean reward prediction	$\bar{x}_i = w_{1,i}\hat{x}_{1,i} + w_{2,i}\hat{x}_{2,i}$	
7	Response rate	$R_i = r_{max}\bar{x}_i / \,(\bar{x}_i + k(1 - \bar{x}_i))$	r_{max} = max. rate k = shape coeff.
8	Mean prediction error	$\bar{\delta}_i = x_i - \bar{x}_i$	
9	Mean uncertainty	$\bar{u}_i = n(\bar{\delta}^2_{i-1} - \bar{u}_{i-1}) + \bar{u}_{i-1}$	
10	Stimulus-reward association	$y_i = \sum_{k=1}^{i-1} 1/\bar{u}_k \bigg/ \left(g + \sum_{k=1}^{i-1} 1/\bar{u}_k\right)$	g = shape coeff.

```
trials = TimedVector(rewards, schedule)
trials

##        time value
##  [1,]     1     1
##  [2,]     2     1
##  [3,]     3     1
##  [4,]     4     1
##  [5,]     5     1
##  [6,]     6     1
##  [7,]     7     1
##  [8,]     8     1
##  [9,]     9     1
## [10,]    10     1
## [11,]  2880     1
## [12,]  2881     1
## [13,]  2882     1
## [14,]  2883     1
## [15,]  2884     1
## [16,]  2885     1
## [17,]  2886     1
## [18,]  2887     1
## [19,]  2888     1
## [20,]  2889     1
## [21,]  5760     0
## [22,]  5761     0
## [23,]  8640     0
## [24,]  8641     0
## [25,] 11520     0
## [26,] 11521     0
## [27,] 14400     0
## [28,] 14401     0
## [29,] 17280     0
## [30,] 17281     0
```

1.2 Computing the model

Next we can compute the model for a given set of parameter values for the previously specified `trials`. Here the parameter values are loosely based on those in Table **??**. By specifying `verbose = TRUE` in the call to `computeModel`, additional information (weights, uncertainties) is returned as attributes in addition to the mean reward prediction. This information can also be ploted.

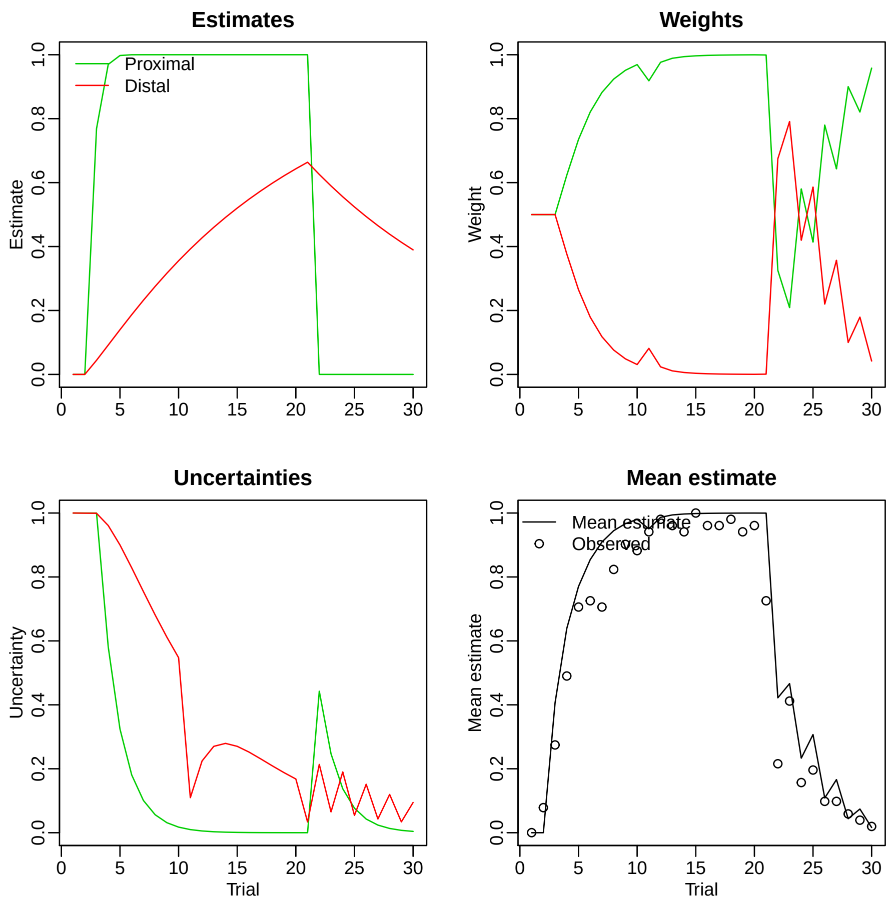
```
params = c(0.9, 0.01, 0.04, 0.4, 0.25, 4.5, 500)
est = computeModel(trials, mFast = params[1], mSlow = params[2], n = params[3],
    h = params[4], g = params[7], verbose = TRUE)
plot(est, actual = rep(NA, length(trials)))

## Warning:  no non-missing arguments to max; returning -Inf
```



1.3 Fitting the model

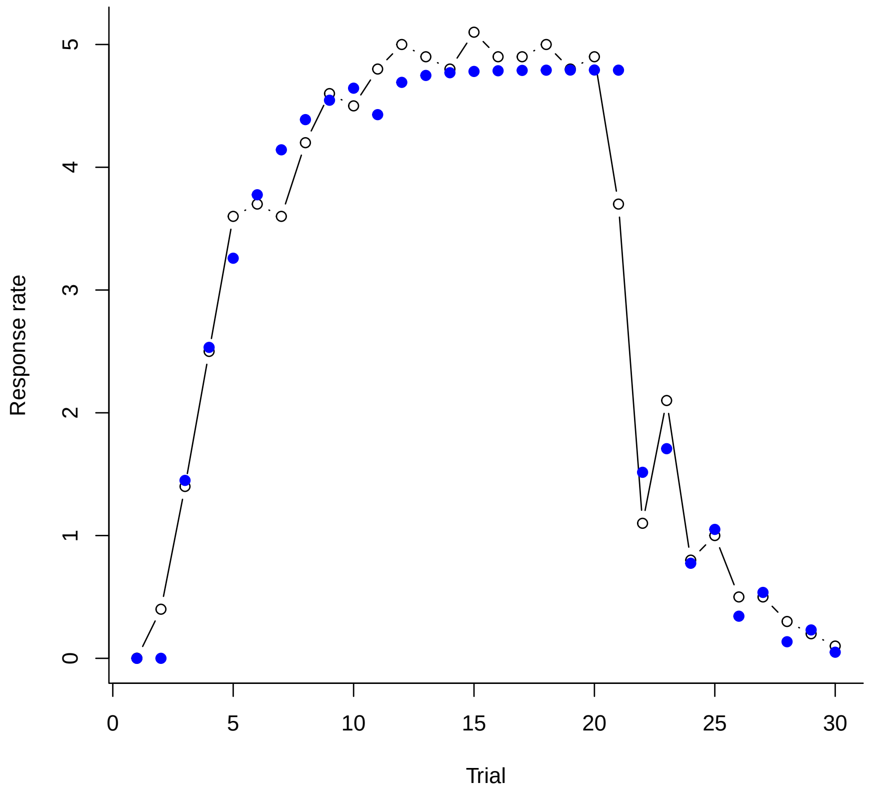
If data on the animal's response rate during the trials is also available, the model's parameters can be estimated from that data.

```
responses = c(0.001, 0.4, 1.4, 2.5, 3.6, 3.7, 3.6, 4.2, 4.6, 4.5, 4.8, 5, 4.9,
    4.8, 5.1, 4.9, 4.9, 5, 4.8, 4.9, 3.7, 1.1, 2.1, 0.8, 1, 0.5, 0.5, 0.3, 0.2,
    0.1)
results = fitModel(dataX = list(trials), dataResponse = list(responses))
fitParams = results$par[1, ]
fitEst = computeModel(trials, mFast = fitParams[1], mSlow = fitParams[2], n = fitParams[3],
    h = fitParams[4], g = fitParams[7], verbose = TRUE)
plot(fitEst, actual = responses/max(responses))
```



In the above diagnostic plots, the response was simply divided by its maximum value in order to place it on the $[0, 1]$ scale of the mean reward prediction. However, since the response function can be non-linear (if $k \neq 1$), the `caluculateResponse` function can be used to compare the predicted response rate to the observed.

```r
fitResponses = calculateResponse(fitParams[5], fitParams[6], fitEst)
plot(1:30, responses, type = "b", bty = "l", xlab = "Trial", ylab = "Response rate")
points(1:30, fitResponses, col = 4, pch = 19)
```



Response rates are frequenly reported for blocks of trials or sessions, and that can be accomplished by specifying how to group trials with `sessionBoundaries`.