Package 'pdmod'

October 14, 2022

Type Package

Title Proximal/Distal Modeling Framework for Pavlovian Conditioning Phenomena

Version 1.0.1

Date 2018-02-12

Author Chloe Bracis

Maintainer Chloe Bracis <cbracis@uw.edu>

Imports mco,stats

Depends

Suggests RUnit

Description Fits a model of Pavlovian conditioning phenomena, such as response extinction and spontaneous recovery, and partial reinforcement extinction effects. Competing proximal and distal reward predictions, computed using fast and slow learning rates, combine according to their uncertainties and the recency of information. The resulting mean prediction drives the response rate.

License GPL (>= 2)

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-13 10:16:37 UTC

R topics documented:

`pdmod-package` *Proximal/Distal Modeling Framework*

Description

In this model, Pavlovian phenomena 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. This package provides methods to compute the model for different parameter values and fit parameters to experimental data.

Details

Package:	pdmod
Type:	Package
Version:	1.0
Date:	2014-03-27
License:	GPL (>=2)

For a given set of rewards/non-rewards paired with a signal in a Pavlovian conditioning experiment (specified as a `TimedVector`), the animal's response for a given set of parameter values can be computed with `computeModel`. Additionally, if experimental response data is available, the parameter values can be fit to the data using `fitModel`. Additional methods `averageBySession` and `plot.pdmod` are available to manipulate and plot model results.

`TimedVector` is a class used to associate reward/no-reward with a time schedule with helper methods `c`, `isTimedVector`, `print`, `time`, and `verifyTimedVector`.

Author(s)

Chloe Bracis

Maintainer: Chloe Bracis <cbracis@uw.edu>

`averageBySession` *Average by session*

Description

Calculates the average estimate per session or block of trials

Usage

```
averageBySession(estimate, sessionBoundaries)
```

Arguments

estimate	Series of estimates in event time
sessionBoundaries	Vector of the starting indices for each session (which means to include the end, the last value should be length(estimate) + 1)

Value

Vector of average estimate for each session

Author(s)

Chloe Bracis

Examples

```
# Create vector of values (i.e. estimates, respones, etc.)
values = runif(100)
# Specify sessions, here a group of 10 trials
sessionBoundaries = seq(1, 101, 10)
valuesBySession = averageBySession(values, sessionBoundaries)
```

---

calculateResponse *Calculate response from the estimate*

Description

Given an estiamtes probability of reward beween 0 and 1, calculates a response rate (i.e. the measured response of the animal such as visits to the food delivery system)

Usage

```
calculateResponse(k, rmax, est)
```

Arguments

k	Response rate parameter
rmax	Maximum response
est	Vector of estimates

Value

Vector of responses

Author(s)

Chloe Bracis

See Also

Constants, isTimedVector, verifyTimedVector

Examples

```
calculateResponse(0.8, 10, runif(20))
```

---

computeModel *Calculates proximal/distal model*

Description

Calulates a realization of a proximal/distal model for a specified sequence of trials and paramter values. Use the verbose parameter to include underlying model components (distal and proximal estimates, weights, uncertainties and signal-reward association) in addition to the mean estimate.

Usage

```
computeModel(x, mFast, mSlow, n, g = 0, h,
  tau = 1/TV_DAY, threshold = 0, verbose = TRUE)
```

Arguments

x	Object of class TimedVector specifying trials including whether signal was rewarded/unrewarded and times
mFast	Learning rate of proximal memory estimates
mSlow	Learning rate of distal memory estimates
n	Learning rate of uncertainty estimates
h	Decay rate of distal memory uncertainty estimator as time passes between trials
g	Association learning speed parameter
tau	Temporal scaling coefficient to translate time differences in x to fractional days. Defaults to 1/TV_DAY assuming that the times in x are expressed in minutes.
threshold	Difference in real time that must pass before deflation kicks in (used for testing)
verbose	true to include supporting estimates, weights, etc.

Value

Series of estimates

Author(s)

Chloe Bracis

See Also

calculateResponse, averageBySession

Examples

```
# Create 5 sessions of 20 rewarded trials,
# then 2 sessions of 20 unrewarded trials
trialTime = as.vector(sapply(0:6, function(x) 1:20 + x * TV_DAY))
trials =  TimedVector(c(rep(1, 5*20), rep(0, 2*20)), trialTime)

estimates = computeModel(trials, mFast = 0.7, mSlow = 0.1, n = 0.05,
 g = 500, h = 0.2, verbose = TRUE)
plot(estimates, trials)
```

Constants *Constants*

Description

Constants to use with TimedVector for specifying time between events.

Usage

```
TV_MINUTE

TV_HOUR

TV_DAY
```

Format

Numeric constants

Details

TV_MINUTE A minute

TV_HOUR An hour

TV_DAY A day

Author(s)

Chloe Bracis

See Also

TimedVector

`fitModel` *Fit model parameters*

Description

Estimates parameters for proximal/distal model using multi-criteria estimation (`mco`)

Usage

```
fitModel(dataX, dataResponse,
  responseFunction = calculateResponse,
  sessionBoundaries = NA, fitG = TRUE)
```

Arguments

`dataX`	Object of class `TimedVector` specifying trials including whether signal was rewarded/unrewarded and times
`dataResponse`	Corresponding observations of subject's response to signal
`responseFunction`	The function to use to transform the mean estimate into a response
`sessionBoundaries`	(optional) Vector defining how to group the trials into sessions where the items are the starting indicies for each session (so the last value can be the index after the last trial) and `NA`s are used for gaps between sessions
`fitG`	`TRUE` (default) to estimate g, or `FALSE` to fix g at 0

Value

Model fit

Author(s)

Chloe Bracis

See Also

`computeModel`

`isTimedVector` *Is TimedVector*

Description

Determines if an object inherits `TimedVector`

Usage

```
isTimedVector(tv)
```

Arguments

`tv`	Potential `TimedVector` object

Value

`TRUE`, if the object inherits `TimedVector` `FALSE`, otherwise

Author(s)

Chloe Bracis

See Also

`TimedVector`, `verifyTimedVector`

Examples

```
# A TimedVector
tv = TimedVector(rep(1, 10), 1:10)
isTimedVector(tv)

# Not a TimedVector
isTimedVector(1:10)
isTimedVector(time(tv))
```

`modelObjectiveFunction`

Objective function to fit model parameters

Description

Function passed to optimization routine to minimize to estimate parameters. Uses mean squared error to calculate difference between `dataResponse` and what `computeModel`) would forcast for `dataX` using parameters `pars`.

Usage

```
modelObjectiveFunction(pars, dimension, dataX,
  dataResponse, responseFunction = calculateResponse,
  sessionBoundaries = NA, fitG = TRUE)
```

Arguments

pars	Vector of parameters mFast, mSlow, n, hSlow, and r
dimension	What dimension to return error in, 1 for single criteria optimization, or number of columns of data for multicriteria optimization
dataX	List of observations of process x(i) (with real time)
dataResponse	Corresponding list of observations of subject's response to x(i), i.e. ~x(i)
responseFunction	The function to use to transform the forecast into a response
sessionBoundaries	(option) Vector defining how to group the trials into sessions where the items are the starting indicies for each session (so the last value can be the index after the last trial) and NAs are used for gaps between sessions
fitG	TRUE to estimate g, or FALSE to fix g at 0

Value

Error between dataRespones and what would have been estimated for dataX based on parameters pars

Author(s)

Chloe Bracis

See Also

computeModel, fitModel

---

plot.pdmod *Plot model*

Description

Plots a pdmod class (what's returned from computeModel with verbose = TRUE). The plots show the proximal and distal estimates, their corresponding uncertainties and weights, as well as the overall mean estimate.

Usage

```
## S3 method for class 'pdmod'
plot(x, actual, n, ...)
```

Arguments

x	Object of class pdmod
actual	Actual rewards received
n	(optional) Only plot the last n values
...	Other arguments to plot

Author(s)

Chloe Bracis

Examples

```
# Create 5 sessions of 20 rewarded trials,
# then 2 sessions of 20 unrewarded trials
trialTime = as.vector(sapply(0:6, function(x) 1:20 + x * TV_DAY))
trials =  TimedVector(c(rep(1, 5*20), rep(0, 2*20)), trialTime)

estimates = computeModel(trials, mFast = 0.7, mSlow = 0.1, n = 0.05,
 g = 500, h = 0.2, verbose = TRUE)
plot(estimates, trials)
```

TimedVector *Create a TimedVector*

Description

The class TimedVector contains a vector of values in event time, as well as when in real time those events took place.

Usage

```
TimedVector(x, t)
```

Arguments

x	Series of values in event time
t	(optional) Cooresponding real time of events in minutes. Default is an event every minute.

Value

TimedVector

Author(s)

Chloe Bracis

See Also

Constants, isTimedVector, verifyTimedVector

Examples

```
# One session of 20 rewarded trials every minute
TimedVector(rep(1, 20), 1:20)

# Three sessions of rewarded trials, then one session of non-rewarded trials,
# with trials every 2 min and sessions every day
trialTime = as.vector(sapply(0:3, function(x) seq(2, 20, 2) + x * TV_DAY))
TimedVector(c(rep(1, 30), rep(0, 10)), trialTime)

# The above schedule of sessions, but 50% probability of reward
TimedVector(sample(0:1, 40, replace = TRUE), trialTime)
```

---

verifyTimedVector *Verify TimedVector*

Description

Verifies object really is a TimedVector (stronger checks than isTimedVector).

Usage

```
verifyTimedVector(tv)
```

Arguments

tv	Potential TimedVector object

Value

TRUE, if the object is a TimedVector FALSE, otherwise

Author(s)

Chloe Bracis

See Also

isTimedVector, TimedVector

Index