

Package ‘catlearn’

December 11, 2017

Type Package

Title Formal Psychological Models of Categorization and Learning

Version 0.5

Date 2017-12-11

Author Andy Wills, Lenard Dome, Charlotte Edmunds, Garrett Honke, Angus Inkster, Rene Schegelmilch, Stuart Spicer

Maintainer Andy Wills <andy@willslab.co.uk>

Description Formal psychological models of categorization and learning, independently-replicated data sets against which to test them, and simulation archives.

License GPL (>= 2)

Imports Rcpp (>= 0.12.6), doParallel, foreach

LinkingTo Rcpp

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-12-11 20:45:54 UTC

R topics documented:

catlearn-package	2
act2probrat	3
nosof88	4
nosof88exalcove	6
nosof88exalcove_opt	7
nosof88oat	8
nosof88protoalcove	9
nosof88protoalcove_opt	10
nosof88train	11
nosof94	13
nosof94bncalcove	15
nosof94exalcove	16

nosof94exalcove_opt	17
nosof94oat	18
nosof94plot	19
nosof94train	20
shin92	22
shin92exalcove	23
shin92exalcove_opt	24
shin92oat	25
shin92protoalcove	26
shin92protoalcove_opt	28
shin92train	29
slpALCOVE	31
slpBM	33
slpCOVIS	36
slpDIVA	40
slpMBMF	42
slpRW	45
ssecl	47
stsimGCM	48
thegrid	52
Index	54

catlearn-package *Formal Modeling for Psychology.*

Description

Formal psychological models, independently-replicated data sets against which to test them, and simulation archives.

Details

For a complete list of functions, use `library(help = "catlearn")`.

For a complete table of simulations, use `data(thegrid)`.

All functions are concisely documented, use the help function e.g `?shin92`.

For more detailed documentation, see the references listed by the help documentation.

For a tutorial introduction, see Wills et al. (2016a).

For a guide to contributing to this package, Catlearn Research Group (2016).

Author(s)

Andy Wills

Maintainer: Andy Wills <andy@willslab.co.uk>

References

Catlearn Research Group (2016). Contributing to catlearn. <http://catlearn.r-forge.r-project.org/intro-catlearn.pdf>

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

act2probrat

Convert output activation to a rating of outcome probability

Description

Logistic function to convert output activations to rating of outcome probability (see e.g. Gluck & Bower, 1988).

Usage

```
act2probrat(act, theta, beta)
```

Arguments

act	Vector of output activations
theta	Scaling constant
beta	Bias constant

Details

The contents of this help file are relatively brief; a more extensive tutorial on using act2probrat can be found in Spicer et al. (n.d.).

The function takes the output activation of a learning model (e.g. slpRW), and converts it into a rating of the subjective probability that the outcome will occur. It does this separately for each activation in the vector act. It uses a logistic function to do this conversion (see e.g. Gluck & Bower, 1988, Equation 7). This function can produce a variety of monotonic mappings from activation to probability rating, determined by the value set for the two constants:

theta is a scaling constant; as its value rises, the function relating activation to rating becomes less linear and at high values approximates a step function.

beta is a bias parameter; it is the value of the output activation that results in an output rating of $P = 0.5$. For example, if you wish an output activation of 0.4 to produce a rated probability of 0.5, set beta to 0.4.

Value

Returns a vector of probability ratings.

Note

As this function returns probabilities, the numbers returned are always in the range 0-1. If the data you are fitting use a different range, convert them. For example, if your data are ratings on a 0-10 scale, divide them by 10. If your data are something other than probability estimates (e.g. you asked participants to use negative ratings to indicate preventative relationships), don't use this function unless you are sure it is doing what you intend.

Author(s)

Andy Wills

References

Gluck, M.A. & Bower, G.H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General*, *117*, 227-247.

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

nosof88

Within-category stimulus frequency CIRP

Description

Within-category stimulus frequency is the number of times a stimulus has been observed as a member of a specific category; sometimes referred to as 'instantiation frequency' (Barsalou, 1985). Increasing instantiation frequency of a stimulus increases categorization accuracy for that stimulus ('direct' effect), and for other similar stimuli ('indirect' effect). Experiment 1 of Nosofsky (1988) provides the data for this CIRP.

Usage

`data(nosof88)`

Format

A data frame with the following columns:

cond Experimental condition, see 'details'. 1 = 'B', 2 = 'E2', 3 = 'E7'

stim Stimulus number, see Nosofsky (1988), Figure 1. Takes values: 1-12

c2acc Mean probability, across participants, of responding that the item belongs to category 2.

Details

Wills & O'Connell (n.d.) discuss the derivation of this CIRP. In brief, both the direct and indirect effects have been independently replicated. Direct effect: Harris et al. (2008), Nosofsky (1988). Indirect effect: Barsalou (1985), Erickson & Kruschke (1998), Florian (1992), Homa et al. (1991), Nosofsky (1991). Experiment 1 of Nosofsky (1988) was selected due to the availability of a multidimensional scaling solution for the stimuli, see [nosof88train](#), and because it includes an assessment of both the direct and the indirect effect.

Experiment 1 of Nosofsky(1988) involved the classification of Munsell chips of fixed hue (5R) varying in brightness (value) and saturation (chroma). Instantiation frequency was manipulated between subjects. In condition B, all stimuli were equally frequent. In condition E2 (E7), stimulus 2 (7) was approximately five times as frequent as each of the other stimuli. In condition E2 (E7), stimulus 4 (9) indexes the indirect effect. There were three blocks of training. Block length was 48 trials for condition B and 63 trials for conditions E2 and E7. The training phase was followed by a transfer phase, which is not included in this CIRP (see Nosofsky, 1988, for details).

The data are as shown in Table 1 of Nosofsky (1988). The data are mean response probabilities for each stimulus in the training phase, averaged across blocks and participants.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Nosofsky, R.M. (1988). Similarity, frequency, and category representations, *Journal of Experimental Psychology: Learning, Memory and Cognition*, 14, 54-65.

References

- Barsalou, L.W. (1985). Ideals, central tendency, and frequency of instantiation as determinants of graded structure in categories. *Journal of Experimental Psychology: Learning, Memory & Cognition*, 11, 629-654.
- Erickson, M.A. & Kruschke, J. (1998). Rules and exemplars in category learning. *Journal of Experimental Psychology: General*, 127, 107-140.
- Florian, J. (1992). Frequency effects of categorization and recognition. *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, 826-831.
- Harris, H.D., Murphy, G.L. & Rehder, B. (2008). Prior knowledge and exemplar frequency. *Memory & Cognition*, 36, 1335-1350.
- Homa, D., Dunbar, S. & Nohre, L. (1991). Instance frequency, categorization, and the modulating effect of experience. *Journal of Experimental Psychology: Learning, Memory & Cognition*, 17, 444-458.
- Nosofsky, R.M. (1991). Tests of an exemplar model for relating perceptual classification and recognition memory. *Journal of Experimental Psychology: Human Perception and Performance*, 17, 3-27.
- Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof88train](#), [nosof88oat](#)

nosof88exalcove

Simulation of CIRP nosof88 with ex-ALCOVE model

Description

Runs a simulation of the [nosof88](#) CIRP using the [slpALCOVE](#) model implementation as an exemplar model and [nosof88train](#) as the input representation.

Usage

```
nosof88exalcove(params = NULL)
```

Arguments

`params` A vector containing values for `c`, `phi`, `la`, and `lw`, in that order, e.g. `params = c(2.1, 0.6, 0.09, 0.9)`. See [slpALCOVE](#) for an explanation of these parameters. Where `params = NULL`, best-fitting parameters are derived from optimization archive [nosof88exalcove_opt](#)

Details

An exemplar-based simulation using [slpALCOVE](#) and [nosof88train](#). The co-ordinates for the radial-basis units are taken from the multidimensional scaling solution for these stimuli reported by Nosofsky (1987).

Other parameters of [slpALCOVE](#) are set as follows: $r = 2$, $q = 1$, initial $\alpha = 1 / (\text{number of input dimensions})$, initial $w = 0$. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the `nosof88exalcove` function, and hence can't be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [nosof88](#).

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13, 87-108.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof88](#), [nosof88oat](#), [nosof88train](#), [slpALCOVE](#)

`nosof88exalcove_opt` *Parameter optimization of ex-ALCOVE model with nosof88 CIRP*

Description

Uses [nosof88exalcove](#) to find best-fitting parameters for the ex-ALCOVE model for the [nosof88 CIRP](#).

Usage

```
nosof88exalcove_opt(recompute = FALSE)
```

Arguments

<code>recompute</code>	When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.
------------------------	--

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the [nosof88exalcove](#) simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the [optim](#) function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. `type nosof88exalcove_opt`). The optimization was repeated for 16 different sets of starting values.

Where `recompute = TRUE`, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virtual cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for c , ϕ , la , and lw , in that order. See [slpALCOVE](#) for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

nosof88oat

Ordinal adequacy test for simulations of nosof88 CIRP

Description

Tests whether a model output passes the ordinal adequacy criteria for the [nosof88](#) CIRP.

Usage

```
nosof88oat(dta, xtdo=FALSE)
```

Arguments

<code>dta</code>	Matrix containing model output. The matrix must have the same format, row order, and column names, as <code>data(nosof88)</code> ; with that proviso, the output of any simulation implementation can be handled by this function.
<code>xtdo</code>	eXTenDed Output: Either TRUE or FALSE

Details

This function implements the Wills & O'Connell (n.d.) ordinal adequacy tests for the [nosof88](#) CIRP. Specifically, a model passes this test if it passes all four component tests: 1. $E2(2) > B(2)$, 2. $E7(7) > B(7)$, 3. $E2(4) > B(4)$, 4. $E7(9) > B(9)$. These tests refer to classification accuracy for particular stimuli in particular experimental conditions. For example, $E7(9)$ indicates stimulus 9 in experimental condition E7.

Alternatively, by setting `xtdo` to TRUE, this function returns the summary model predictions reported by Wills & O'Connell (2016).

Value

Where `xtdo=FALSE`, this function returns TRUE if the ordinal adequacy tests are passed, and FALSE otherwise.

Where `xtdo=TRUE`, this function returns a summary matrix. The columns are stimulus numbers. The rows ('B','E') indicate the baseline (equal frequency) condition ('B') and the experimental conditions ('E2' or 'E7', depending on the column).

Author(s)

Andy Wills and Garret O'Connell

References

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof88](#)

`nosof88protoalcove` *Simulation of CIRP nosof88 with proto-ALCOVE model*

Description

Runs a simulation of the [nosof88](#) CIRP using the [slpALCOVE](#) model implementation as a prototype model and [nosof88train](#) as the input representation.

Usage

```
nosof88protoalcove(params = NULL)
```

Arguments

`params` A vector containing values for `c`, `phi`, `la`, and `lw`, in that order, e.g. `params = c(2.1, 0.6, 0.09, 0.9)`. See [slpALCOVE](#) for an explanation of these parameters. Where `params = NULL`, best-fitting parameters are derived from optimization archive [nosof88protoalcove_opt](#)

Details

An prototype-based simulation using [slpALCOVE](#) and [nosof88train](#). There is one radial-basis unit for each category, representing the prototype. These prototypes are calculated by taking the mean of the co-ordinates of the stimuli in a category, with the stimulus co-ordinates coming from the multidimensional scaling solution reported by Nosofsky (1987). The calculations of the means are weighted by the instantiation frequency of the stimuli. Hence, the prototypes for each condition of the experiment are different.

Other parameters of [slpALCOVE](#) are set as follows: $r = 2$, $q = 1$, initial $\alpha = 1 / (\text{number of input dimensions})$, initial $w = 0$. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the [nosof88protoalcove](#) function, and hence can not be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [nosof88](#).

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13, 87-108.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof88](#), [nosof88oat](#), [nosof88train](#), [slpALCOVE](#)

`nosof88protoalcove_opt`

Parameter optimization of proto-ALCOVE model with nosof88 CIRP

Description

Uses [nosof88protoalcove](#) to find best-fitting parameters for the ex-ALCOVE model for the [nosof88](#) CIRP.

Usage

```
nosof88protoalcove_opt(recompute = FALSE)
```

Arguments

recompute When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the [nosof88protoalcove](#) simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the [optim](#) function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type `nosof88protoalcove_opt`). The optimization was repeated for 16 different sets of starting values.

Where `recompute = TRUE`, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virtual cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for `c`, `phi`, `la`, and `lw`, in that order. See [slpALCOVE](#) for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

nosof88train	<i>Input representation of nosof88 for models input-compatible with slpALCOVE.</i>
--------------	--

Description

Create randomized training blocks for CIRP [nosof88](#), in a format suitable for the [slpALCOVE](#) model, and any other model that uses the same input representation format. The stimulus co-ordinates come from a MDS solution reported by Nosofsky (1987) for the same stimuli.

Usage

```
nosof88train(condition = 'B', blocks = 3, absval = -1, subjs = 1, seed =
4182, missing = 'geo')
```

Arguments

condition	Experimental condition 'B', 'E2', or 'E7', as defined by Nosofsky (1988).
blocks	Number of blocks to generate. Omit this argument to get the same number of blocks as the published study (3).
absval	Teaching value to be used where category is absent.
subjs	Number of simulated subjects to be run.
seed	Sets the random seed
missing	If set to 'geo', output missing dimension flags (see below)

Details

A matrix is produced, with one row for each trial, and with the following columns:

ctrl - Set to 1 (reset model) for trial 1, set to zero (normal trial) for all other trials.

cond - 1 = condition B, 2 = condition E2, 3 = condition E7

blk - training block

stim - stimulus number (as defined by Nosofsky, 1988)

x1, x2 - input representation. These are the co-ordinates of an MDS solution for these stimuli (see Nosofsky, 1987).

t1, t2 - teaching signal (1 = category present, absval = category absent)

m1, m2 - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Only produced if missing = 'geo'.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This implementation assumes a block length of 64 trials for conditions E2 and E7, rather than the 63 trials reported by Nosofsky (1988).

This routine was originally developed to support simulations reported in Wills & O'Connell (n.d.).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition*, 13, 87-108.

Nosofsky, R.M. (1988). Similarity, frequency, and category representations, *Journal of Experimental Psychology: Learning, Memory and Cognition*, 14, 54-65.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof88](#), [nosof88oat](#), [slpALCOVE](#)

nosof94

Type I-VI category structure CIRP

Description

Shepard et al. (1961) stated that where there are two, equal-sized categories constructed from the eight stimuli it is possible to produce from varying three binary stimulus dimensions, there are only six logically distinct category structures. Shepard et al. (1961) labeled these structures as Types I through VI (see e.g. Nosofsky et al., 1994, Figure 1, for details). The CIRP concerns the relative difficulty of learning these category structures, as indexed by classification accuracy. The result, expressed in terms of accuracy, is:

$I > II > [III, IV, V] > VI$

The experiment reported by Nosofsky et al. (1994) provides the data for this CIRP.

Usage

```
data(nosof94)
```

Format

A data frame with the following columns:

type Type of category structure, as defined by Shepard et al. (1961). Takes values : 1-6

block Training block. Takes values: 1-16

error Mean error probability, averaged across participants

Details

Wills & O'Connell (n.d.) discuss the derivation of this CIRP. In brief, the effect has been demonstrated independently on three occasions (Nosofsky et al., 1994; Shepard et al., 1961; Smith et al., 2004). Kurtz et al. (2013) discuss and report a number of experiments that indicate that the advantage of Type II over Type IV may depend on specific details of the experimental design, including the stimulus dimensions being easily verbalizable, and the participants being explicitly instructed to look for rules.

Nosofsky et al. (1994) was selected as the CIRP because it had acceptable sample size (N=40 per Type), and included simulations of the results with a number of different formal models. Inclusion of this dataset in `cat.learn` thus permits a validation of `cat.learn` model implementations against published simulations.

In Nosofsky et al. (1994) the stimuli varied in shape (squares or triangles), type of interior line (solid or dotted), and size (large or small). Each participant learned two problems. Each problem was trained with feedback, to a criterion of four consecutive sub-blocks of eight trials with no errors, or for a maximum of 400 trials.

The data are as shown in the first 16 rows of Table 1 of Nosofsky et al. (1994). Only the first 16 blocks are reported, for comparability with the model fitting reported in that paper. Where a participant reached criterion before 16 blocks, Nosofsky et al. assumed they would have made no further errors if they had continued.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352-369.

References

Kurtz, K.J., Levering, K.R., Stanton, R.D., Romero, J. & Norris, S.N. (2013). Human learning of elemental category structures: Revising the classic result of Shepard, Hovland, and Jenkins (1961). *Journal of Experimental Psychology: Learning, Memory & Cognition*, 39, 552-572.

Shepard, R.N., Hovland, C.I., & Jenkins, H.M. (1961). learning and memorization of classifications. *Psychological Monographs*, 75, Whole No. 517.

Smith, J.D., Minda, J.P. & Washburn, D.A. (2004). Category learning in rhesus monkeys: A study of the Shepard, Hovland, and Jenkins (1961) tasks. *Journal of Experimental Psychology: General*, 133, 398-414.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof94train](#), [nosof94oat](#)

 nosof94bncove

Simulation of CIRP nosof94 with BN-ALCOVE model

Description

Runs a simulation of the [nosof94](#) CIRP using the [slpALCOVE](#) model implementation as an exemplar model and [nosof94train](#) as the input representation. This simulation replicates the one reported by Nosofsky et al. (1994).

Usage

```
nosof94bncove(params = c(6.33, 0.011, 0.409, 0.179))
```

Arguments

`params` A vector containing values for `c`, `phi`, `la`, and `lw`, in that order. See [slpALCOVE](#) for an explanation of these parameters.

Details

An exemplar-based simulation using [slpALCOVE](#) and [nosof94train](#). The co-ordinates for the radial-basis units are assumed, and use the same binary representation as the abstract category structure.

The defaults for `params` are the best fit of the model to the [nosof94](#) CIRP. The derivation of this fit is described by Nosofsky et al. (1994).

The other parameters of [slpALCOVE](#) are set as follows: $r = 1$, $q = 1$, initial `alpha` = 1 / number of dimensions, initial `w` = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the [nosof88bncove](#) function, and hence can't be changed without re-writing the function.

This is a replication of the simulation reported by Nosofsky et al. (1994). Compared to other published simulations with the ALCOVE model, their simulation is non-standard in a number of respects:

1. A background noise ('BN') decision rule is used (other simulations use an exponential ratio rule).
2. As a consequence of #1, absence of a category label is represented by a zero (other simulations use -1).
3. The sum of the attentional weights is constrained to be 1 on every trial (other simulations do not apply this constraint).

The current simulation replicates these non-standard aspects of the Nosofsky et al. (1994) simulation.

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [nosof94](#).

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352–369

See Also

[nosof94](#), [nosof94oat](#), [nosof94train](#), [slpALCOVE](#), [nosof94bnalcove](#)

nosof94exalcove

Simulation of CIRP nosof94 with ex-ALCOVE model

Description

Runs a simulation of the [nosof94](#) CIRP using the [slpALCOVE](#) model implementation as an exemplar model and [nosof94train](#) as the input representation.

Usage

```
nosof94exalcove(params = NULL)
```

Arguments

params A vector containing values for *c*, *phi*, *la*, and *lw*, in that order, e.g. `params = c(2.1, 0.6, 0.09, 0.9)`. See [slpALCOVE](#) for an explanation of these parameters. Where `params = NULL`, best-fitting parameters are derived from optimization archive [nosof94exalcove_opt](#)

Details

N.B.: This simulation uses a standard version of ALCOVE. For a replication of the ALCOVE simulation of these data reported by Nosofsky et al. (1994), which is non-standard in a number of respects, see [nosof94bnalcove](#).

An exemplar-based simulation using [slpALCOVE](#) and [nosof94train](#). The co-ordinates for the radial-basis units are assumed, and use the same binary representation as the abstract category structure.

Other parameters of [slpALCOVE](#) are set as follows: $r = 1$, $q = 1$, initial $\alpha = 1/3$, initial $w = 0$. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the `nosof88exalcove` function, and hence can't be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [nosof94](#).

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352–369

Wills, A.J. & O’Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof94](#), [nosof94oat](#), [nosof94train](#), [slpALCOVE](#), [nosof94bnalcove](#)

`nosof94exalcove_opt` *Parameter optimization of ex-ALCOVE model with nosof94 CIRP*

Description

Uses [nosof94exalcove](#) to find best-fitting parameters for the ex-ALCOVE model for the [nosof94 CIRP](#).

Usage

```
nosof94exalcove_opt(recompute = FALSE, xtdo = FALSE)
```

Arguments

<code>recompute</code>	When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.
<code>xtdo</code>	eXTenDed Output; where set to TRUE, some further details of the optimization procedure are printed to the console.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the [nosof94exalcove](#) simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the [optim](#) function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type `nosof94exalcove_opt`). The optimization was repeated for 15 different sets of starting values.

Where `recompute = TRUE`, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virtual cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for `c`, `phi`, `la`, and `lw`, in that order. See [slpALCOVE](#) for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

nosof94oat

Ordinal adequacy test for simulations of nosof94 CIRP

Description

Tests whether a model output passes the ordinal adequacy criteria for the [nosof94](#) CIRP.

Usage

```
nosof94oat(dta, xtdo=FALSE)
```

Arguments

dta	Matrix containing model output. The matrix must have the same format, row order, and column names, as <code>data(nosof94)</code> ; with that proviso, the output of any simulation implementation can be handled by this function.
xtdo	eXTenDed Output: Either TRUE or FALSE

Details

This function implements a standard ordinal adequacy test for the `nosof94` CIRP. Specifically, a model passes this test if the mean errors (averaged across blocks), obey the following:

$$I < II < [III, IV, V] < VI$$

Note that '[III, IV, V]' indicates that these three problems can be in any order of difficulty (or all be of equal difficulty), as long as all three are harder than Problem 2 and all three are easier than Problem 6.

Alternatively, by setting `xtdo` to TRUE, this function returns the mean classification error by Problem type.

Value

Where `xtdo=FALSE`, this function returns TRUE if the ordinal adequacy tests are passed, and FALSE otherwise.

Where `xtdo=TRUE`, this function returns a summary matrix, containing mean errors (across blocks) for each of the six problem types.

Author(s)

Andy Wills

See Also

[nosof94](#)

`nosof94plot`

Plot Nosofsky et al. (1994) data / simulations

Description

Produce a line graph similar to that shown in Nosofsky et al. (1994, Figures 1, 6-9).

Usage

```
nosof94plot(results, title = 'Nosofsky et al. (1994)')
```

Arguments

results	Mean error probability by block and problem, in the same format as data set nosof94
title	Title to appear at top of plot

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352–369.

Examples

```
data(nosof94)
nosof94plot(nosof94)
```

nosof94train	<i>Input representation of nosof94 for models input-compatible with slpALCOVE.</i>
--------------	--

Description

Create randomized training blocks for CIRP nosof94, in a format suitable for the slpALCOVE model, and other models that use the same input representation format. The stimulus co-ordinates are assumed, and use the same binary representation as the abstract category structure.

Usage

```
nosof94train(cond = 1, blocks = 16, absval = -1, subjs = 1, seed = 7624,
missing = 'geo')
```

Arguments

cond	Category structure type (1-6), as defined by Shepard et al. (1961).
blocks	Number of blocks to generate. Omit this argument to get the same number of blocks (16) as used in the simulations reported by Nosofsky et al. (1994).
absval	Teaching value to be used where category is absent.
subjs	Number of simulated subjects to be run.
seed	Sets the random seed.
missing	If set to 'geo', output missing dimension flags (see below)

Details

A matrix is produced, with one row for each trial, and with the following columns:

`ctr1` - Set to 1 (reset model) for trial 1, set to zero (normal trial) for all other trials.

`blk` - training block

`stim` - stimulus number (arbitrary, but consistent across different values of problem)

`x1`, `x2`, `x3` - input representation

`t1`, `t2` - teaching signal (1 = category present, absval = category absent)

`m1`, `m2`, `m3` - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Only produced if `missing = 'geo'`.

In Nosofsky et al. (1994), block 1 was randomized differently to later blocks. This feature is retained in this implementation.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the `seed` argument to set a different seed.

This routine was originally developed to support simulations reported in Wills & O'Connell (n.d.).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352–369

Shepard, R.N., Hovland, C.I., & Jenkins, H.M. (1961). learning and memorization of classifications. *Psychological Monographs*, 75, Whole No. 517.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

[nosof94train](#), [nosof94oat](#)

shin92

Category size CIRP

Description

Category size is the number of examples of a category that have been presented to the participant. The category-size effect (e.g. Homa et al., 1973) is the phenomenon that, as category size increases, the accuracy of generalization to new members of that category also increases. The equal-frequency conditions of Experiment 3 of Shin & Nosofsky (1992) provides the data for this CIRP.

Usage

```
data(shin92)
```

Format

A data frame with the following columns:

catsize Experimental condition (category size). Takes values : 3, 10

cat Category membership of stimulus. Takes values: 1, 2

stim Stimulus code, as defined by Shin & Nosofsky (1992). Stimuli beginning 'RN' or 'URN' are the 'novel' stimuli. Stimuli beginning 'P' are prototypes. The remaining stimuli are the 'old' (training) stimuli.

c2acc Mean probability, across participants, of responding that the item belongs to category 2.

Details

Wills et al. (2016) discuss the derivation of this CIRP. In brief, a between-subjects manipulation of category size was selected to avoid a potential response-bias confound in within-subject studies of the category-size effect. The between-subjects effect has been demonstrated independently on three occasions (DiNardo & Toppino, 1984; Homa et al., 1987; Shin & Nosofsky, 1992). Experiment 3 of Shin & Nosofsky (1992) was selected due to the availability of a multi-dimensional scaling solution for the stimuli, see [shin92train](#).

Experiment 3 of Shin & Nosofsky (1992) involved the classification of nine-vertex polygon stimuli drawn from two categories. Category size was manipulated between subjects (3 vs. 10 stimuli per category). Participants received eight blocks of training, and three test blocks.

The data are as shown in Table 10 of Shin & Nosofsky (1992). The data are mean response probabilities for each stimulus in the test phase, averaged across test blocks and participants.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, 121, 278-304.

References

- DiNardo, M.J. & Toppino, T.C. (1984). Formation of ill-defined concepts as a function of category size and category exposure. *Bulletin of the Psychonomic Society*, 22, 317-320.
- Homa, D., Burrue, L. & Field, D. (1987). The changing composition of abstracted categories under manipulations of decisional change, choice difficulty, and category size. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 401-412.
- Homa, D., Cross, J., Cornell, D., Goldman, D. & Schwartz, S. (1973). Prototype abstraction and classification of new instances as a function of number of instances defining the prototype. *Journal of Experimental Psychology*, 101, 116-122.
- Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

See Also

[shin92train](#), [shin92oat](#)

shin92exalcove

Simulation of CIRP shin92 with ex-ALCOVE model

Description

Runs a simulation of the [shin92](#) CIRP using the [slpALCOVE](#) model implementation as an exemplar model and [shin92train](#) as the input representation.

Usage

```
shin92exalcove(params = NULL)
```

Arguments

`params` A vector containing values for `c`, `phi`, `la`, and `lw`, in that order, e.g. `params = c(2.1, 0.6, 0.09, 0.9)`. See [slpALCOVE](#) for an explanation of these parameters. Where `params = NULL`, best-fitting parameters are derived from optimization archive [shin92exalcove_opt](#)

Details

An exemplar-based simulation using [slpALCOVE](#) and [shin92train](#). The co-ordinates for the radial-basis units are derived from the test stimuli in [shin92train](#). The output is the average of 100 simulated subjects.

The defaults for `params` are the best fit of the model to the [shin92](#) CIRP. They were derived through minimization of SSE using non-linear optimization from 16 different initial states (using code not included in this archive).

The other parameters of `slpALCOVE` are set as follows: $r = 2$, $q = 1$, initial $\alpha = 1 /$ (number of input dimensions), initial $w = 0$. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the `shin92exalcove` function, and hence can't be changed without re-writing the function.

This simulation was reported in Wills et al. (2017).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [shin92](#).

Author(s)

Andy Wills & Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, 121, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*, 66, 79-115.

shin92exalcove_opt *Parameter optimization of ex-ALCOVE model with shin92 CIRP*

Description

Uses `shin92exalcove` to find best-fitting parameters for the ex-ALCOVE model for the [shin92 CIRP](#).

Usage

```
shin92exalcove_opt(params = c(2, 1, 0.25, 0.75), recompute = FALSE,
trace = 0)
```

Arguments

params	A vector containing the initial values for c , ϕ , l_a , and l_w , in that order. See slpALCOVE for an explanation of these parameters. Where <code>recompute</code> is <code>FALSE</code> , this argument has no effect.
recompute	When set to <code>TRUE</code> , the function re-runs the optimization (which takes about 25 minutes on a 2.4 GHz processor). When set to <code>FALSE</code> , the function returns a stored copy of the results of the optimization (which is instantaneous).

`trace` Sets the level of tracing information (i.e. information about the progress of the optimization), as defined by the `optim` function. Set to 6 for maximally verbose output. Where `recompute` is `FALSE`, this argument has no effect.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the `shin92exalcove` simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the `optim` function of the standard R `stats` package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type `shin92exalcove_opt`).

This function was run in 16 times from different starting points, using 8 threads on a Core i7 3.6 GHz processor. The default parameters of this function are those for the best fit from those 16 starting points. The 16 starting points were

```
pset <- rbind( c(2,1,.25,.25),c(2,1,.25,.75),c(2,1,.75,.25),c(2,1,.75,.75), c(2,3,.25,.75),
               )
```

not all of which converged successfully.

Value

A vector containing the best-fitting values for `c`, `phi`, `la`, and `lw`, in that order. See `slpALCOVE` for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

shin92oat

Ordinal adequacy test for simulations of shin92 CIRP

Description

Tests whether a model output passes the ordinal adequacy criterion for the `shin92` CIRP.

Usage

```
shin92oat(dta, xtdo=FALSE)
```

Arguments

dta	Matrix containing model output. The matrix must have the same format, row order, and column names, as that returned by <code>shin92exalcove</code> ; with that proviso, the output of any simulation implementation can be handled by this function.
xtdo	eXTenDed Output: Either TRUE or FALSE

Details

This function implements the Wills et al. (2016) ordinal adequacy test for the `shin92` CIRP. Specifically, a model passes this test if response accuracy is higher for novel items from the size-10 condition than novel items from the size-3 condition.

Alternatively, by setting `xtdo` to TRUE, this function returns the summary model predictions reported by Wills et al. (2016).

Value

Where `xtdo=FALSE`, this function returns TRUE if the ordinal adequacy test is passed, and FALSE otherwise.

Where `xtdo=TRUE`, this function returns a summary matrix. The rows are the two category sizes, the columns are the three principal stimulus types (old, prototype, new), and the values are predicted accuracy scores.

Author(s)

Andy Wills and Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, 121, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

See Also

[shin92](#)

shin92protoalcove *Simulation of CIRP shin92 with proto-ALCOVE model*

Description

Runs a simulation of the `shin92` CIRP using the `slpALCOVE` model implementation as a prototype model and `shin92train` as the input representation.

Usage

```
shin92protoalcove(params = NULL)
```

Arguments

params A vector containing values for *c*, *phi*, *la*, and *lw*, in that order, e.g. `params = c(2.1, 0.6, 0.09, 0.9)`. See [slpALCOVE](#) for an explanation of these parameters. Where `params = NULL`, best-fitting parameters are derived from optimization archive [shin92exalcove_opt](#)

Details

An exemplar-based simulation using [slpALCOVE](#) and [shin92train](#). The co-ordinates for the radial-basis units for the two prototypes are derived from the arithmetic means of the test stimuli in [shin92train](#). The output is the average of 100 simulated subjects.

The defaults for `params` are the best fit of the model to the [shin92](#) CIRP. They were derived through minimization of SSE using non-linear optimization from 16 different initial states (using code not included in this archive).

The other parameters of [slpALCOVE](#) are set as follows: $r = 2$, $q = 1$, initial $\alpha = 1 / (\text{number of input dimensions})$, initial $w = 0$. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the `shin92exalcove` function, and hence can't be changed without re-writing the function.

This simulation was reported in Wills et al. (2017).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in [shin92](#).

Author(s)

Andy Wills & Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, *121*, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*, *66*, 79-115.

shin92protoalcove_opt *Parameter optimization of proto-ALCOVE model with shin92 CIRP*

Description

Uses `shin92protoalcove` to find best-fitting parameters for the proto-ALCOVE model for the `shin92` CIRP.

Usage

```
shin92protoalcove_opt(params = c(2,1,.25,.75), recompute = FALSE,
  trace = 0)
```

Arguments

<code>params</code>	A vector containing the initial values for <code>c</code> , <code>phi</code> , <code>la</code> , and <code>lw</code> , in that order. See <code>slpALCOVE</code> for an explanation of these parameters. Where <code>recompute</code> is <code>FALSE</code> , this argument has no effect.
<code>recompute</code>	When set to <code>TRUE</code> , the function re-runs the optimization (which takes about 10 minutes on a 2.4 GHz processor). When set to <code>FALSE</code> , the function returns a stored copy of the results of the optimization (which is instantaneous).
<code>trace</code>	Sets the level of tracing information (i.e. information about the progress of the optimization), as defined by the <code>optim</code> function. Set to 6 for maximally verbose output. Where <code>recompute</code> is <code>FALSE</code> , this argument has no effect.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the `shin92protoalcove` simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the `optim` function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type `shin92protoalcove_opt`).

This function was run in 16 times from different starting points, using 8 threads on a Core i7 3.6 GHz processor. The default parameters of this function are those for the best fit from those 16 starting points. The 16 starting points were

```
pset <- rbind(
  c(2,1,.25,.25),c(2,1,.25,.75),c(2,1,.75,.25),c(2,1,.75,.75),
  c(2,3,.25,.25),
  )
```

not all of which converged successfully.

Value

A vector containing the best-fitting values for `c`, `phi`, `la`, and `lw`, in that order. See `slpALCOVE` for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

shin92train	<i>Input representation of shin92 for models input-compatible with slpALCOVE.</i>
-------------	---

Description

Creates randomized training and transfer blocks for CIRP shin92 , in a format suitable for the slpALCOVE model, and any other model that uses the same input representation format. The stimulus co-ordinates come from a MDS solution reported by Shin & Nosofsky (1992).

Usage

```
shin92train(condition = 'equal3', learn.blocks = 8, trans.blocks = 3,
            absval = -1, format = 'mds', subjs = 1, seed = 8416, missing =
            'geo')
```

Arguments

condition	Experimental condition 'equal3', 'equal10', 'unequal3', or 'unequal10', as defined by Shin & Nosofsky (1992).
learn.blocks	Number of training blocks to generate. Omit this argument to get the same number of training blocks as the published study (8).
trans.blocks	Number of transfer blocks to generate. Omit this argument to get the same number of transfer blocks as the published study (3).
absval	Teaching value to be used where category is absent.
format	Specifies format used for input representation. Only one format is currently supported, so this option is provided solely to support future development.
subjs	Number of simulated subjects to be run.
seed	Sets the random seed
missing	If set to 'geo', output missing dimension flags (see below)

Details

A matrix is produced, with one row for each trial, and with the following columns:

ctr1 - Set to 1 (reset model) for trial 1, set to zero (normal trial) for all other training trials, and set to 2 (freeze learning) for all transfer trials.

cond - 1 = equal3, 2 = equal10, 3 = unequal3, 4 = unequal10

phase - 1 = training, 2 = transfer

blk - block of trials

stim - stimulus number; these correspond to the rows in Tables A3 and A4 of Shin & Nosofsky (1992)

x1 ... x6 - input representation. These are the co-ordinates of an MDS solution for these stimuli (see Shin & Nosofsky, 1992, Tables A3 and A4). Note: Size 3 conditions have a four-dimensional MDS solution, so the output is x1 ... x4

t1, t2 - teaching signal (1 = category present, absval = category absent)

m1 ... m6 - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Note: ranges from m1 to m4 for Size 3 conditions. Only produced if missing = 'geo'.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This function was originally developed to support simulations reported in Wills et al. (2017).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, 121, 278-304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*, 66.

See Also

[shin92](#), [shin92oat](#), [slpALCOVE](#)

slpALCOVE

*ALCOVE category learning model***Description**

Kruschke's (1992) category learning model.

Usage

```
slpALCOVE(st, tr, dec = 'ER', humble = TRUE, attcon = FALSE, absval = -1,
          xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
dec	String defining decision rule to be used
humble	Boolean specifying whether a humble or strict teacher is to be used
attcon	Boolean specifying whether attention is constrained
absval	Real number specifying teaching value for category absence
xtdo	Boolean specifying whether to write extended information to the console (see below).

Details

The coverage in this help file is relatively brief; Catlearn Research Group (2016) provides an introduction to the mathematics of the ALCOVE model, whilst a more extensive tutorial on using slpALCOVE can be found in Wills et al. (2016).

The functions works as a stateful list processor. Specifically, it takes a matrix as an argument, where each row is one trial for the network, and the columns specify the input representation, teaching signals, and other control signals. It returns a matrix where each row is a trial, and the columns are the response probabilities at the output units. It also returns the final state of the network (attention and connection weights), hence its description as a 'stateful' list processor.

Argument `st` must be a list containing the following items:

`colskip` - skip the first `N` columns of the `tr` array, where `N = colskip`. `colskip` should be set to the number of optional columns you have added to matrix `tr`, PLUS ONE. So, if you have added no optional columns, `colskip = 1`. This is because the first (non-optional) column contains the control values, below.

`c` - specificity constant (Kruschke, 1992, Eq. 1). Positive real number. Scales psychological space.

`r` - distance metric (Kruschke, 1992, Eq. 1). Set to 1 (city-block) or 2 (Euclidean).

`q` - similarity gradient (Kruschke, 1992, Eq. 1). Set to 1 (exponential) or 2 (Gaussian).

ϕ - decision constant. For decision rule ER, it is referred to as mapping constant ϕ , see Kruschke (1992, Eq. 3). For decision rule BN, it is referred to as the background noise constant b , see Nosofsky et al. (1994, Eq. 3).

l_w - associative learning rate (Kruschke, 1992, Eq. 5) . Real number between 0 and 1.

l_a - attentional learning rate (Kruschke, 1992, Eq. 6). Real number between 0 and 1.

h - R by C matrix of hidden node locations in psychological space, where R = number of input dimensions and C = number of hidden nodes.

α - vector of length N giving initial attention weights for each input dimension, where N = number of input dimensions. If you are not sure what to use here, set all values to 1.

w - R by C matrix of initial associative strengths, where R = number of output units and C = number of hidden units. If you are not sure what to use here, set all values to zero.

Argument tr must be a matrix, where each row is one trial presented to the network. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

$ctrl$ - vector of control codes. Available codes are: 0 = normal trial, 1 = reset network (i.e. set attention weights and associative strengths back to their initial values as specified in h and w (see below)), 2 = Freeze learning. Control codes are actioned before the trial is processed.

$opt1$, $opt2$, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the $ctrl$ column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument $colskip$ (see above) must be set to the number of optional columns plus 1.

$x1$, $x2$, ... - input to the model, there must be one column for each input unit. Each row is one trial.

$t1$, $t2$, ... - teaching signal to model, there must be one column for each output unit. Each row is one trial. If the stimulus is a member of category X , then the teaching signal for output unit X must be set to +1, and the teaching signal for all other output units must be set to $absval$.

$m1$, $m2$, ... - missing dimension flags, there must be one column for each input unit. Each row is one trial. Where $m = 1$, that input unit does not contribute to the activation of the hidden units on that trial. This permits modelling of stimuli where some dimensions are missing on some trials (e.g. where modelling base-rate neglect, Kruschke, 1992, p. 29–32). Where $m = 0$, that input unit contributes as normal. If you are not sure what to use here, set to zero.

Argument dec , if specified, must take one of the following values:

ER specifies an exponential ratio rule (Kruschke, 1992, Eq. 3).

BN specifies a background noise ratio rule (Nosofsky et al., 1994, Eq. 3). Any output activation lower than zero is set to zero before entering into this rule.

Argument $humble$ specifies whether a humble or strict teacher is to be used. The function of a humble teacher is specified in Kruschke (1992, Eq. 4b). In this implementation, the value -1 in Equation 4b is replaced by $absval$.

Argument $attcon$ specifies whether attention should be constrained or not. *If you are not sure what to use here, set to FALSE.* Some implementations of ALCOVE (e.g. Nosofsky et al., 1994) constrain the sum of the attentional weights to always be 1 (personal communication, R. Nosofsky,

June 2015). The implementation of attentional constraint in *alcovel* is the same as that used by Nosofsky et al. (1994), and present as an option in the source code available from Kruschke's website (Kruschke, 1991).

Argument `xtdo` (eXTenDed Output), if set to TRUE, will output to the console the following information on every trial: (1) trial number, (2) attention weights at the end of that trial, (3) connection weights at the end of that trial, one row for each output unit. This output can be quite lengthy, so diverting the output to a file with the `sink` command prior to running *alcovel* with extended output is advised.

Value

Returns a list containing three components: (1) matrix of response probabilities for each output unit on each trial, (2) attentional weights after final trial, (3) connection weights after final trial.

Author(s)

Andy Wills

References

- Catlearn Research Group (2016). Description of ALCOVE. <http://catlearn.r-forge.r-project.org/desc-alcove.pdf>
- Kruschke, J. (1991). *ALCOVE.c*. Retrieved 2015-07-20, <http://www.indiana.edu/~kruschke/articles/ALCOVE.c>
- Kruschke, J. (1992). ALCOVE: an exemplar-based connectionist model of category learning. *Psychological Review*, 99, 22-44
- Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, 22, 352-369.
- Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation*, 66, 79-115.

slpBM

Bush & Mosteller (1951) simple associative learning model

Description

A model often attributed to Bush & Mosteller (1951), more precisely this is the separable error term learning equation discussed by authors such as Mackintosh (1975) and Le Pelley (2004); see Note 1.

Usage

`slpBM(st, tr, xtdo = FALSE)`

Arguments

<code>st</code>	List of model parameters
<code>tr</code>	R matrix of training items
<code>xtdo</code>	Boolean specifying whether to include extended information in the output (see below)

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (`tr`) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence or absence of an outcome. It returns the output activation on each trial (a.k.a sum of associative strengths of cues present on that trial), as a vector. The `slpBM` function also returns the final state of the model - a vector of associative strengths between each stimulus and the outcome representation.

Argument `st` must be a list containing the following items:

`lr` - the learning rate (fixed for a given simulation), as denoted by, for example, θ in Equation 1 of Mackintosh (1975). If you want different elements to differ in salience (different alpha values) use the input activations (`x1`, `x2`, ..., see below) to represent element-specific salience.

`w` - a vector of initial associative strengths. If you are not sure what to use here, set all values to zero.

`colskip` - the number of optional columns to be skipped in the `tr` matrix. `colskip` should be set to the number of optional columns you have added to the `tr` matrix, PLUS ONE. So, if you have added no optional columns, `colskip=1`. This is because the first (non-optional) column contains the control values (details below).

Argument `tr` must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set associative strengths (weights) back to their initial values as specified in `w` (see above)); 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1`, `opt2`, ... - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument `colskip` (see above) must be set to the number of optional columns plus one.

`x1`, `x2`, ... - activation of any number of input elements. There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, `slpBM` supports any real number for activations, e.g. one might use values between 0 and 1 to represent differing cue saliences.

t - Teaching signal (a.k.a. lambda). Traditionally, 1 is used to represent the presence of the outcome, and 0 is used to represent the absence of the outcome, although slpBM supports any real values for lambda..

Argument xtdo (eXTenDed Output) - if set to TRUE, function will return the associative strengths for the end of each trial (see Value).

Value

Returns a list containing two components (if xtdo = FALSE) or three components (if xtdo = TRUE, xout is also returned):

st	Vector of final associative strengths
suma	Vector of output activations for each trial
xout	Matrix of associative strengths at the end of each trial

Note

1. Bush & Mosteller's (1951) Equations 2 outputs response probability, not associative strength. Also, it has two learning rate paramters, a and b. At least to a first approximation, b serves a similar function to beta-outcome-absent in Rescorla & Wagner (1972), and a-b is similar to beta-outcome-present in that same model.

Author(s)

Lenard Dome, Stuart Spicer, Andy Wills

References

- Bush, R. R., & Mosteller, F. (1951). A mathematical model for simple learning. *Psychological Review*, 58(5), 313-323.
- Kamin, L.J. (1969). Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.
- Le Pelley, M.E. (2004). The role of associative history in models of associative learning: A selective review and a hybrid model, *Quarterly Journal of Experimental Psychology*, 57B, 193-243.
- Mackintosh, N.J. (1975). A theory of attention: Variations in the associability of stimuli with reinforcement, *Psychological Review*, 82, 276-298.
- Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-99). New York: Appleton-Century-Crofts.
- Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.
- Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation*, 66, 79-115.

slpCOVIS

*COVIS category learning model***Description**

COmpetition between Verbal and Implicit Systems model of category learning (Ashby et al. 1998), as described in Ashby et al. (2011). The current implementation supports two-category experiments, and uses only single-dimension, not-below-chance, rules in the Explicit system.

Usage

```
slpCOVIS(st, tr, crx = TRUE, respt = FALSE, rgive = TRUE, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
crx	Boolean. Explicit System. If set to TRUE, the current rule is included in the random selection of a rule to receive a weight increase from the Poisson distribution. If set to FALSE, the current rule is not included in this random selection.
respt	Set to FALSE for the behaviour described in Note 5; behaviour when TRUE is undocumented
rgive	Set to TRUE; FALSE is undocumented
xtdo	Set to FALSE; TRUE is undocumented

Details

The coverage in this help file is relatively brief; for a more extensive tutorial, see Inkster et al. (n.d.).

The function works as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row is one trial for the network, and the columns specify the input representation. It returns a List containing the predictions made by the model and the final state of the model, hence its description as a 'stateful' list processor.

Argument st must be a list containing the following information. Parameter names given in brackets in the descriptions below follow the naming conventions of Ashby et al. (2011), and Edmunds & Wills (2016). Equation numbers are from Ashby et al. (2011); where there is no equation, the page number is given instead.

Explicit system variables:

envar - (σ^2_E) - p. 68 - Variance of the noise distribution used to determine which response the explicit system makes on the current trial. See Note 4, below.

decbound - (C) - Eq. 1 - location of the decision bound on a single dimension. In the current implementation of slpCOVIS, this location is the same for all dimensions.

corcon - (δ_c) - Eq. 2 - constant by which to increase current rule saliency in the case of a correct response.

errcon - (δ_e) - Eq. 3 - constant by which to decrease current rule saliency in the case of an incorrect response.

perscon - (γ) - Eq. 4 - perseveration constant, i.e. value to add to the saliency of the current rule to obtain its rule weight.

lambda - (λ) - Eq. 5 - Mean of the Poisson distribution. A value randomly sampled from the Poisson distribution is added to a randomly-selected rule when calculating the weights for new rule selection.

decsto - (a) - Eq. 7 - decision stochasticity when using rule weights to select the rule for the next trial. For Ashby et al. (2011)'s implementation, $a = 1$. For other uses, see Edmunds & Wills (2016).

Procedural system variables:

sconst - (α) - Eq. 8 - scaling constant for cortical unit activation. See Note 3, below.

invar - (σ^2_p) - Eq. 9 - Variance of the normally-distributed noise used to calculate striatal unit activation.

dbase - (D_{base}) - Eq. 10 - baseline dopamine level.

alphaw - (α_w) - Eq. 10 - Learning rate parameter in force when striatal activation is above the NMDA threshold, and dopamine is above baseline.

betaw - (β_w) - Eq. 10 - Learning rate parameter in force when striatal activation is above the NMDA threshold, and dopamine is below baseline.

gammax - (γ_w) - Eq. 10 - Learning rate parameter in force when striatal activation is between the AMPA and NMDA thresholds.

nmda - (θ_{NMDA}) - Eq. 10 - Activation threshold for post-synaptic NMDA.

ampa - (θ_{AMPA}) - Eq. 10 - Activation threshold for post-synaptic AMPA. See Note 1, below.

wmax - (w_{max}) - Eq. 10 - Intended upper weight limit for a cortico-striatal link. See Note 2, below.

prep - ($P_{(n-1)}$) - Eq. 12 - predicted reward value immediately prior to first trial. If unsure, set to zero.

prer - ($R_{(n-1)}$) - Eq. 12 - obtained reward value immediately prior to first trial. If unsure, set to zero.

Competition / decision system variables:

emaxval - p.77 - The maximum possible value of the the Explicit system's discriminant variable. For example, if the stimulus value varies from zero to one, and C (see above) is 0.5, then the maximum value is $1-0.5 = 0.5$

etrust - (θ_E) - Eq. 15 - trust in the explicit system immediately prior to first trial. If unsure, set to .99.

itrust - (θ_P) - p. 77 - trust in the procedural system immediately prior to first trial. If unsure, set to .01. See also Note 7, below.

ocp - (δ_{OC}) - Eq. 15 - constant used to increase trust in the Explicit system after it suggests a response that turns out to be correct.

oep - (δ_{OE}) - Eq. 16 - constant used to decrease trust in the Explicit system after it suggests a response that turns out to be incorrect.

Initial state of model:

initrules - vector of length *stindim*, representing the initial salience of each single-dimensional rule in the Explicit system.

crule - a number indicating which rule is in use immediately prior to the first trial (1 = dimension 1, 2 = dimension 2, etc). If this is not meaningful in the context of your simulation, set it to zero, and ensure *ctrl* = 1 in the first row of your training matrix (see below). This will then randomly pick an initial rule.

initsy - matrix of *stindim* rows and two columns - contains the initial values for the cortico-striatal connection strengths.

scups - matrix of *stindim* columns and as many rows as you wish to have cortical input units. Each row represents the position of a cortical unit in N-dimensional stimulus space.

And finally, a couple of things slpCOVIS needs to interpret your *tr* matrix (see below):

stindim - number of stimulus dimensions in the input representation.

colskip - skip the first N columns of the *tr* array, where N = *colskip*. *colskip* should be set to the number of optional columns you have added to matrix *tr*, PLUS ONE. So, if you have added no optional columns, *colskip* = 1. This is because the first (non-optional) column contains the control values, see below.

Argument *tr* must be a matrix, where each row is one trial presented to the network. Trials are always presented to the model in the order specified. The columns must be as described below, in the order described below:

ctrl - vector of control codes. Available codes are: 0 = normal trial, 1 = reset network (i.e. set back to the state defined in list *st* and randomly select an initial rule for the Explicit System using Eq. 7), 2 = Freeze learning. Control codes are actioned before the trial is processed.

opt1, *opt2*, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the *ctrl* column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument *colskip* (see above) must be set to the number of optional columns plus 1.

x1, *x2*, ... - stimulus input to the model; there must be one column for each stimulus dimension.

t1 - teaching signal to model. If the correct response is Category 1, *t* = 1. If the correct response is Category 2, *t* = -1. Experiments with something other than two categories are not supported in the current implementation.

optend1, *optend2*, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the *t1* column. These optional columns are ignored by this function, but may help with cross-compatibility with other model implementations. For example, the additional 't' and 'm' columns of input representations generated for slpALCOVE will be safely ignored by slpCOVIS.

Value

Returns a List containing eight components:

foutmat A two-column matrix, representing the model's response on each trial. For any given trial, [1,0] indicates a Category 1 response; [0,1] indicates a Category 2

	response. Responses are reported in this manner to facilitate cross-compatibility with models that produce response probabilities on each trial.
frules	Explicit system - rule saliences after final trial
fsystr	Procedural system - cortico-striatal synaptic strengths after final trial)
fetrust	Decision system - trust in explicit system after final trial
fitrust	Decision system - trust in procedural system after final trial
frule	Explicit system - rule used by explicit system on final trial
fprep	Implicit system - predicted reward value on final trial
fprer	Implicit system - obtained reward value on final trial

Note

1. Ashby et al. (2011) state (p. 74) that the intended operation of COVVIS is $\theta_{\text{NMDA}} > \theta_{\text{AMPA}}$, but the values they report are $\theta_{\text{NMDA}} = .0022$, $\theta_{\text{AMPA}} = .01$.
2. Ashby et al. (2011) did not specify a value for w_{max} ; Edmunds & Wills (2016) assumed the intended value was 1.
3. Ashby et al. (2011) do not use Eq. 8 in their simulation, they manually set sensory cortex activation to 1 for the presented stimulus and 0 for all the others (p. 78). They thus do not have a value for α . Edmunds & Wills (2016) set α to 0.14, which produces similar behaviour for 0,1 coded stimulus dimensions, without having to manually set the activations.
4. In Ashby et al. (2011) and Edmunds & Wills (2016), σ^2_{E} is set to zero. In this implementation of slpRW, positive values should also work but have not been extensively tested.
5. In the descriptions provided by Ashby et al. (2011, p. 69 & p. 75), there is some ambiguity about the meaning of the term 'response' - does this mean the response of a system (e.g. the Explicit system), or the overall response (i.e. the output of the decision system). In the current implementation, the response of the Explicit System is compared to the feedback to determine whether the Explicit System was correct or incorrect, and the response of the Procedural System is compared to the feedback to determine whether the Procedural System was correct or incorrect.
6. It seems that in Ashby et al.'s (2011) simulations, each dimension generates only one single-dimension rule for a two-category problem, rather than two as one might expect (e.g. small = A, large = B, but also large = A, small = B). Rules that would produce below-chance responding are excluded from the rule set.
7. Ashby et al. (2011) state that $\theta_{\text{E}} + \theta_{\text{P}} = 1$. However, slpCOVIS does not perform this check on the initial state, so it is important to check this manually.

Author(s)

Angus Inkster, Andy Wills, Charlotte Edmunds

References

- Ashby, F.G., Alfonso-Reese, L.A., Turken, A.U. & Waldron, E.M. (1998). A neuropsychological theory of multiple systems in category learning. *Psychological Review*, *105*, 442-481.
- Ashby, F.G., Paul, E.J., & Maddox, W.T. (2011). COVIS. In Pothos, E.M. & Wills, A.J. (2011). *Formal approaches in categorization*. Cambridge, UK: Cambridge University Press.

Edmunds, C.E.R., & Wills, A.J. (2016). Modeling category learning using a dual-system approach: A simulation of Shepard, Hovland and Jenkins (1961) by COVIS. In A. Papfragou, D. Grodner, D. Mirman, & J.C. Trueswell (Eds.). *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (pp. 69-74). Austin, TX: Cognitive Science Society.

Inkster, A.B., Edmunds, C.E.R., & Wills, A.J. (n.d.). A distributed-collaboration resource for dual-process modeling in category learning. *Manuscript in preparation*.

Pothos, E.M. & Wills, A.J.(2011). *Formal approaches in Categorisation*.Cambridge: University Press.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation*, 66, 79-115.

 slpDIVA

DIVA category learning model

Description

DIVergent Autoencoder (Kurtz, 2007; 2015) artificial neural network category learning model

Usage

```
slpDIVA(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
xtdo	When set to TRUE, produce extended output

Details

This function works as a stateful list processor (Wills et al., 2017). Specifically, it takes a matrix as an argument, where each row is one trial for the network, and the columns specify the input representation, teaching signals, and other control signals. It returns a matrix where each row is a trial, and the columns are the response probabilities for each category. It also returns the final state of the network (connection weights and other parameters), hence its description as a 'stateful' list processor.

Argument `st` must be a list containing the following items:

`st` must contain the following principal model parameters:

`learning_rate` - Learning rate for weight updates through backpropagation. The suggested learning rate default is `learning_rate = 0.15`

`beta_val` - Scalar value for the Beta parameter. `beta_val` controls the degree of feature focusing (not unlike attention) that the model uses to make classification decisions (see: Conaway & Kurtz, 2014; Kurtz, 2015). `beta_val = 0` turns feature focusing off.

`phi` - Scalar value for the phi parameter. phi is a real-valued mapping constant, see Kruschke (1992, Eq. 3).

`st` must also contain the following information about network architecture:

`num_feats` - Number of input features.

`num_hids` - Number of hidden units. A rough rule of thumb for this hyperparameter is to start with `num_feats = 2` and add additional units if the model fails to converge.

`num_cats` - Number of categories.

`continuous` - A Boolean value to indicate if the model should work in continuous input or binary input mode. Set `continuous = TRUE` when the inputs are continuous.

`st` must also contain the following information about the initial state of the network:

`in_wts` - A matrix of initial input-to-hidden weights with `num_feats + 1` rows and `num_hids` columns. Can be set to NULL when the first line of the `tr` matrix includes control code 1, `ctrl = 1`.

`out_wts` - A matrix of initial hidden-to-output weights with `num_feats + 1` rows, `num_hids` columns and with the third dimension being `num_cats` in extent. Can be set to NULL when the first line of the `tr` matrix includes control code 1, `ctrl = 1`.

`st` must also contain the following information so that it can reset these weights to random values when `ctrl = 1` (see below):

`wts_range` - A scalar value for the range of the randomly-generated weights. The suggested weight range default is `wts_range = 1`

`wts_center` - A scalar value for the center of the randomly-generated weights. This is commonly set to `wts_center = 0`

`st` must also contain the following parameters that describe your `tr` array:

`colskip` - Skip the first N columns of the `tr` array, where `N = colskip`. `colskip` should be set to the number of optional columns you have added to matrix `tr`, PLUS ONE. So, if you have added no optional columns, `colskip = 1`. This is because the first (non-optional) column contains the control values, below.

Argument `tr` must be a matrix, where each row is one trial presented to the network. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - column of control codes. Available codes are: 0 = normal learning trial, 1 = reset network (i.e. initialize a new set of weights following the `st` parameters), 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1`, `opt2`, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the `ctrl` column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument `colskip` (see above) must be set to the number of optional columns plus 1.

`x1`, `x2`, ... - input to the model, there must be one column for each input unit. Each row is one trial. Dichotomous inputs should be in the format `-1, 1`. Continuous inputs should be scaled to the range of `-1, 1`. As the model's learning objective is to accurately reconstruct the inputs, the input to the model is also the teaching signal. For testing under conditions of missing information, input

features can be set to 0 to negate the contribution of the feature(s) for the classification decision of that trial.

t_1 , t_2 , ... - Category membership of the current stimulus. There must be one column for each category. Each row is one trial. If the stimulus is a member of category X, then the value in the category X column must be set to +1, and the values for all other category columns must be set to -1.

Value

Returns a list containing two components: (1) matrix of response probabilities for each category on each trial, (2) an `st` list object that contains the model's final state. A weight initialization history is also available when the extended output parameter is set `xtdo = TRUE` in the `slpDIVA` call.

Note

A faster (Rcpp) implementation of `slpDIVA` is planned for a future release of `catlearn`.

Author(s)

Garrett Honke, Nolan B. Conaway, Andy Wills

References

- Conaway, N. B., & Kurtz, K. J. (2014). Now you know it, now you don't: Asking the right question about category knowledge. In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of the Thirty-Sixth Annual Conference of the Cognitive Science Society* (pp. 2062-2067). Austin, TX: Cognitive Science Society.
- Kruschke, J. (1992). ALCOVE: an exemplar-based connectionist model of category learning. *Psychological Review*, *99*, 22-44
- Kurtz, K.J. (2007). The divergent autoencoder (DIVA) model of category learning. *Psychonomic Bulletin & Review*, *14*, 560-576.
- Kurtz, K. J. (2015). Human Category Learning: Toward a Broader Explanatory Account. *Psychology of Learning and Motivation*, *63*.
- Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*, *66*, 79-115.

slpMBMF

MB/MF reinforcement learning model

Description

Gillan et al.'s (2015) model-free / model-based hybrid Reinforcement Learning model (see Note 1).

Usage

`slpMBMF(st, tr, xtdo = FALSE)`

Arguments

st	List of model parameters
tr	Matrix of training items
xtdo	Boolean. When TRUE, extended output is provided, see below

Details

The contents of this help file are relatively brief; a more extensive discussion of this model can be found in the supplementary materials of Gillan et al. (2015).

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model. It returns a matrix of predicted response probabilities for each stage 1 action on each trial. The slpMBMF function also returns the final Q values for the model.

The current implementation of slpMBMF deals only with relatively simple Reinforcement Learning experiments, of which Gillan et al. (2015, Exp. 2) is one example. Specifically, each trial has two stages. In the first stage of the trial, there is a single state, and the participant can emit one of x actions. In the second stage, there are y states. A reward follows (or doesn't) without a further action from the participant.

A hybrid MB/MF model thus has $2x$ Q-values at stage 1 (x for the model-based system, and x for the model-free system), and y Q-values at stage 2 (one for each state; there are no actions at stage 2, and the MB and MF systems evaluate stage 2 Q-values the same way in this model). See Note 3.

Argument st must be a list containing the following items:

alpha - the model-free learning rate (range: 0-1)

lambda - the eligibility trace parameter (range: 0-1)

w - A number between 0 and 1, representing the relative contribution of the model-based and model-free parts of the model to the response (0 = pure model-free, 1 = pure model-based).

beta - Decision stochasticity parameter

p - Decision perseveration ($p > 0$) or switching ($p < 0$) parameter

tprob - A 2×2 matrix of transition probabilities, used by the model-based system. The rows are the actions at stage 1. The columns are the states at stage 2. The cells are transition probabilities (e.g. tprob[2,1] is the probability of arriving at stage 2 state #1 given action #2 at stage 1).

q1.mf - A vector of initial model-free Q values for the actions at stage 1.

q1.mb - A vector of initial model-based Q values for the actions at stage 1.

q2 - A vector of initial Q values for the states at stage 2 (the MB and MF systems share common Q values at stage 2).

If you are unsure what initial Q values to use, set all to 0.5.

Argument tr must be a matrix, where each row is one trial. Trials are always presented to the model in the order specified. The matrix must contain the following named columns (other columns will be ignored):

s1.act - The action made by the participant at stage 1, for each trial; must be an integer in the range 1- x .

`s2.state` - State of environment at stage 2, for each trial; must be an integer in the range 1-y.

`t` - Reward signal for trial; must be a real number. If you're unsure what to use here, use 1 = rewarded, 0 = not rewarded.

Value

When `xtdo = FALSE`, returns a list containing these components:

`out` - Matrix of response probabilities, for each stage 1 action on each trial.

`q1.mf` - A vector of final model-free Q values for the actions at stage 1.

`q1.mb` - A vector of final model-based Q values for the actions at stage 1

`q2` - A vector of final Q values for the states at stage 2 (the MB and MF systems share common Q values at stage 2).

When `xtdo = TRUE`, the list also contains the following model-state information :

`xout` - A matrix containing the state of the model at the end of each trial. Each row is one trial. It has the following columns:

`q1.mb.1`, `q1.mb.2`, ... - One column for each model-based Q value at stage 1.

`q1.mf.1`, `q1.mf.2`, ... - One column for each model-free Q value at stage 1.

`q2.1`, `q2.2`, ... - One column for each Q value at stage 2.

`q1.h.1`, `q1.h.2`, ... - One column for each hybrid Q value at stage 1.

`s1.d.mf` - Model-free delta at stage 2, wrt. stage 1 action.

`s2.d.mf` - Model-free delta at outcome.

In addition, when `xtdo = TRUE`, the list also contains the following information that is not used by the model (but which might be handy as potential neural regressors).

`s1.d.mb` - Model-based delta at stage 2, wrt. stage 1 action.

`s1.d.h` - Hybrid delta (based on stage 1 hybrid Q values) at stage 2, wrt. stage 1 action.

`s1.d.diff` - `s1.d.mf` - `s1.d.mb`

Note

1. Gillan et al.'s (2015) choice rule, at least as stated in their supplementary materials, would lead to the response probabilities being infinite on switch trials, which is presumably an error. The current implementation uses Daw et al. (2011, suppl. mat., Eq. 2).

2. Gillan et al. (2015) decay Q values for unselected actions by (1-alpha). This is not part of the current implementation.

3. In the current implementation of the model, `x` must be 2 and `y` must be two, otherwise the model will fail or behave unpredictably. If you'd like to develop a more general version of this implementation, contact the author.

Author(s)

Andy Wills (andy@willslab.co.uk), Tom Sambrook

References

- Daw, N.D., Gershman, S.J., Seymour, B., Dayan, P., & Dolan, R.J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, *69*, 1204-1215.
- Gillan, C.M., Otto, A.R., Phelps, E.A. & Daw, N.D. (2015). Model-based learning protects against forming habits. *Cogn. Affect. Behav. Neurosci.*, *15*, 523-536.
- Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling thrX-Sough distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation*, *66*, 79-115.

 slpRW

Rescorla-Wagner (1972) associative learning model.

Description

Rescorla & Wagner's (1972) theory of Pavlovian conditioning.

Usage

```
slpRW(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	Matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below)

Details

The contents of this help file are relatively brief; a more extensive tutorial on using slpRW can be found in Spicer et al. (n.d.).

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence/absence of an outcome. It returns the output activation on each trial (a.k.a. sum of associative strengths of cues present on that trial), as a vector. The slpRW function also returns the final state of the model - a vector of associative strengths between each stimulus and the outcome representation.

Argument st must be a list containing the following items:

lr - the learning rate (fixed for a given simulation). In order to calculate lr, calculate the product of Rescorla-Wagner parameters alpha and beta. For example, if you want alpha = 0.1 and beta = 0.2, set lr = 0.02. If you want different elements to differ in salience (different alpha values) use the input activations (x1, x2, ..., see below) to represent element-specific salience. For example, if

alpha_A = 0.4, alpha_X = 0.2, and beta = 0.1, then set lr = 0.1, and the activations of A and B to 0.4 and 0.2, respectively.

w - a vector of initial associative strengths. If you are not sure what to use here, set all values to zero.

colskip - the number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument tr must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

ctrl - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set associative strengths (weights) back to their initial values as specified in w (see above)); 2 = Freeze learning. Control codes are actioned before the trial is processed.

opt1, opt2, ... - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the slpRW function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

x1, x2, ... - activation of any number of input elements. There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, slpRW supports any real number for activations, e.g. one might use values between 0 and 1 to represent differing cue saliences.

t - Teaching signal (a.k.a. lambda). Traditionally, 1 is used to represent the presence of the outcome, and 0 is used to represent the absence of the outcome, although slpRW supports any real values for lambda.

Argument xtdo (eXTenDed Output) - if set to TRUE, function will return associative strength for the end of each trial (see Value).

Value

Returns a list containing two components (if xtdo = FALSE) or three components (if xtdo = TRUE, xout is also returned):

suma	Vector of output activations for each trial
st	Vector of final associative strengths
xout	Matrix of associative strengths at the end of each trial

Author(s)

Stuart Spicer, Lenard Dome, Andy Wills

References

Kamin, L.J., (1969) Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-99). New York: Appleton-Century-Crofts.

Spicer, S.G., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation*, 66, 79-115.

ssecl

Sum of squared errors

Description

Calculate sum of squared errors

Usage

`ssecl(obs, exp)`

Arguments

<code>obs</code>	Vector of observed values
<code>exp</code>	Vector of expected values

Value

Returns sum of the squared differences.

Author(s)

Andy Wills

stsimGCM

*Generalized Context Model***Description**

Nosofsky's (1984, 2011) Generalized Context Model; an exemplar-based model of categorization.

Usage

```
stsimGCM(st)
```

Arguments

`st` List of model parameters

Details

Argument `st` must be a list containing the following required items: `training_items`, `tr`, `nCats`, `nFeat`, `sensitivity`, `weights`, `choice_bias`, `p`, `r_metric`, `mp`, and `gamma`

`nCats` - integer indicating the number of categories

`nFeat` - integer indicating the number of stimulus dimensions

`tr` - the stimuli presented to the model, for which the choice probabilities will be predicted. `tr` has to be a matrix or dataframe with one row for each stimulus. `tr` requires the following columns.

`x1`, `x2`, ... - columns for each dimension carrying the corresponding values (have to be coded as numeric values) for each exemplar (trial) given in the row. Columns have to start with `x1` ascending with dimensions ..., `x2`, `x3`, ... at adjacent columns.

`tr` may have any number of additional columns with any desired name and position, e.g. for readability. As long as the feature columns `x1`, `x2`, ... are given as defined (i.e. not scattered, across the range of matrix columns), the output is not affected by optional columns.

`training_items` - all unique exemplars assumed to be stored in memory; has to be a matrix or dataframe with one row for each exemplar. The rownames have to start with 1 in ascending order. `training_items` requires the following columns:

`x1`, `x2`, ... - columns for each feature dimension carrying the corresponding values (have to be coded as numeric values) for each exemplar (row). Columns have to start with `x1` ascending with dimensions ..., `x2`, `x3`, ... at adjacent columns.

`cat1`, `cat2`, ... - columns that indicate the category assignment of each exemplar (row). For example, if the exemplar in row 2 belongs to category 1 the corresponding cell of `cat1` has to be set to 1, else 0. Columns have to start with `cat1` ascending with categories ..., `cat2`, `cat3`, ... at adjacent columns.

`mem` - (optional) one column that indicates whether an exemplar receives an extra memory weight, `yes = 1`, `no = 0`. For each exemplar (row) in the `training_items` with `mem` set to 0 the corresponding memory strength parameter is set to 1. When `mem` for an exemplar is set to 1 the memory strength parameter is set as defined in `mp`, see below.

training_items may have any number of additional columns with any desired name and position, e.g. for readability. As long as the feature columns x_1 , x_2 , ... and cat_1 , cat_2 , ... are given as defined (i.e. not scattered, across the range of matrix columns), the output is not affected by optional columns.

NOTE: The current model can be implemented as a prototype model if the training_items only carry one row for each category representing the values of the corresponding prototypes (e.g. see Minda & Smith, (2011).

mp - memory strength parameter (optional). Can take any numeric value between -Inf and +Inf. The default is 1, i.e. all exemplars have the same memory strength. There are two ways of specifying mp, i.e. either *globally* or *exemplar specific*:

When *globally* setting mp to a single integer, e.g. to 5, then all exemplars in training_items with $mem = 1$ will receive a memory strength 5 times higher than the memory strengths for the remaining exemplars.

For setting *exemplar specific* memory strengths mp has to be a vector of length n, where n is the overall number of exemplars with $mem = 1$ in the training_items. The order of memory strengths defined in this vector exactly follows their row-wise ascending order of appearance in the training_items. E.g. if there are two exemplars with $mem = 1$ in the training_items, the first one in row 2 and the second one in row 10, then setting mp to $c(3,2)$ will result in assigning a memory strength of 3 to the first exemplar (in row 2) and a memory strength of 2 to the second exemplar (in row 10). The memory strengths for all other exemplars will be set to 1. See Note 1.

sensitivity - sensitivity parameter c; can take any value between 0 (all exemplars are equally similar) and +infinity (towards being insensitive to large differences). There are two ways of specifying sensitivity, i.e. either *globally* or *exemplar specific*: When *globally* setting sensitivity to a single value, e.g. $sensitivity=3$, then the same parameter is applied to all exemplars. On the other hand, *exemplar specific* sensitivity parameters can be used by defining sensitivity as a vector of length n, where n is the number of rows in training_items. The sensitivity vector values then represent the sensitivity parameters for all exemplars in training_items at the corresponding row positions. E.g. if there are 3 exemplars (rows) in training_items, then setting sensitivity to $c(1,1,3)$ assigns $sensitivity = 1$ to the first two exemplars, and $sensitivity = 3$ for the third exemplar. See Note 2.

weights - dimensional attention weights. Order corresponds to the definitions of x_1 , x_2 , ... in tr and training_items. Has to be a vector with length n-1, where n equals to nFeat dimension weights, e.g. of length 2 when there are three features, leaving out the *last* dimension. A constraint in the GCM is that all attentional weights sum to 1. Thus, the sum of n-1 weights should be equal to or smaller than 1, too. The last n-th weight then is computed within the model with: $1 - (\text{sum of } n-1 \text{ feature weights})$. When setting the weights to $1/nFeat = \text{equal weights}$. See Note 3.

choice_bias - Category choice biases. Has to be a vector with length n-1, where n equals to nCats category biases, leaving out the last category bias, under the constraint that all biases sum to 1. Order corresponds to the definitions of cat_1 , cat_2 in the training_items. The sum of n-1 choice biases has to be equal to or smaller than 1. Setting the weights to $1/nCats = \text{no choice bias}$. The bias for the last category then is computed in the model with: $1 - (\text{sum of } nCats-1 \text{ choice biases})$. See Note 3.

gamma - decision constant/ response scaling. Can take any value between 0 (towards more probabilistic) and +infinity (towards deterministic choices). Nosofsky (2011) suggests setting gamma higher than 1 when individual participants' data are considered. See Note 2.

r_metric - distance metric. Set to 1 (city-block) or 2 (Euclidean). See Nosofsky (2011), and Note 4, for more details.

p - similarity gradient. Set to 1 (exponential) or 2 (Gaussian). See Nosofsky (2011), for more details.

Value

A matrix of probabilities for category responses (columns) for each stimulus (rows) presented to the model (e.g. test trials). Stimuli and categories are in the same order as presented to the model in *st*, see below.

Note

1. Please note that setting $mp = 1$ or e.g. $mp = 5$ globally, will yield identical response probabilities. Crucially, memory strength is indifferent from the category choice bias parameter, if (and only if) mp 's vary between categories, without varying within categories. Thus, the memory strength parameter can therefore be interpreted in terms of an exemplar choice bias (potentially related to categorization accuracy). In addition, if exemplar specific mp 's are assigned during parameter fitting, one might want to calculate the natural log of the corresponding estimates, enabling direct comparisons between mp 's indicating different directions, e.g. $-\log(.5) = \log(2)$, for loss and gain, respectively, which are equal regarding their extent into different directions.

2. Theoretically, increasing global sensitivity indicates that categorization mainly relies on the most similar exemplars, usually making choices less probabilistic. Thus sensitivity c is likely to be correlated with γ . See Navarro (2007) for a detailed discussion. However, it is possible to assume exemplar specific sensitivities, or specificity. Then, exemplars with lower sensitivity parameters will have a stronger impact on stimulus similarity and thus categorization behavior for stimuli. See Rodrigues & Murre (2007) for a related study.

3. Setting only the $n-1$ instead of all n feature weights (or bias parameters) eases model fitting procedures, in which the last weight always is a linear combination of the $n-1$ weights.

4. See Tversky & Gati (1982) for further info on r . In brief summary, $r=2$ (usually termed Euclidean), then a large difference on only one feature outweighs small differences on all features. In contrast, if $r=1$ (usually termed City-Block or Manhattan distance) both aspects contribute to an equal extent to the distance. Thus, $r = 2$ comes with the assumption that small differences in all features may be less recognized, than a large noticeable differences on one feature, which may depend on confusability of the stimuli or on the nature of the given task domain (perceptual or abstract).

Author(s)

Rene Schlegelmilch, Andy Wills

References

Minda, J. P., & Smith, J. D. (2011). Prototype models of categorization: Basic formulation, predictions, and limitations. *Formal approaches in categorization*, 40-64.

Navarro, D. J. (2007). On the interaction between exemplar-based concepts and a response scaling process. *Journal of Mathematical Psychology*, 51(2), 85-98.

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, memory, and cognition*, 10(1), 104.

Nosofsky, R. M. (2011). The generalized context model: An exemplar model of classification. In Pothos, E.M. & Wills, A.J. *Formal approaches in categorization*. Cambridge University Press.

Rodrigues, P. M., & Murre, J. M. (2007). Rules-plus-exception tasks: A problem for exemplar models?. *Psychonomic Bulletin & Review*, 14(4), 640-646.

Tversky, A., & Gati, I. (1982). Similarity, separability, and the triangle inequality. *Psychological review*, 89(2), 123.

Examples

```
## Three Categories with 2 Training Items each, and repeatedly presented
## transfer/test items (from nosof94train()). Each item has three
## features with two (binary) values: memory strength (st$mp and
## 'mem' column in st$training_items are optional) is
## equal for all exemplars

st<-list(
  sensitivity = 3,
  weights = c(.2,.3),
  choice_bias = c(1/3),
  gamma = 1,
  mp = 1,
  r_metric = 1,
  p = 1,
  nCats = 2,
  nFeat=3
)

## training item definitions
st$training_items <- as.data.frame(
  t(matrix(cbind(c(1,0,1,1,1,0,0),c(1,1,0,2,1,0,0),
    c(0,1,0,5,0,1,0),c(0,0,1,1,0,1,0)),
    ncol=4, nrow=7,
    dimnames=list(c("stim","x1", "x2", "x3",
      "cat1", "cat2", "mem"),
      c(1:4))))))

st$tr <- nosof94train()

## get the resulting predictions for the test items

## columns of the output correspond to category numbers as defined
## above rows correspond to the column indices of the test_items

stsimGCM(st)

## columns of the output correspond to category numbers as defined
## above rows correspond to the column indices of the test_items
```

```

## Example 2

## Same (settings) as above, except: memory strength is 5 times higher
## for for some exemplars
st$mp<-5

## which exemplars?
## training item definitions
st$training_items <- as.data.frame(
  t(matrix(cbind(c(1,0,1,1,1,0,1),c(1,1,0,2,1,0,0),
                c(0,1,0,5,0,1,0),c(0,0,1,1,0,1,1)),
          ncol=4, nrow=7,
          dimnames=list(c("stim","x1", "x2", "x3",
                          "cat1", "cat2", "mem"),
                        c(1:4))))))
## exemplars in row 1 and 4 will receive a memory strength of 5
## get predictions
stsimGCM(st)

## Example 3
## Same (settings) as above, except: memory strength is item specific
## for the two exemplars i.e. memory strength boost is not the same
## for both exemplars (3 for the first in row 1, and 5 for the
## second exemplar in row 4)
st$mp<-c(3,5)

## get predictions
stsimGCM(st)

```

 thegrid

Ordinal adequacy results for all catlearn simulations

Description

Records results of all ordinal adequacy tests registered in the catlearn package.

Usage

```
data(thegrid)
```

Format

A data frame with the following columns:

- id** Unique identifier number for each entry into the grid. When making a new entry, use the next available integer.
- cirp** The CIRP (Canonical Independently Replicated Phenomenon) against which a model was tested. This must correspond precisely to the name of a data set in the catlearn package.

model A one-word description of the model being tested. Simulations in the same row of The Grid must have precisely the same one-word description. Note, this is not the name of the function used to run the simulation, nor the name of the model implementation function. It is a descriptive term, defined by the modeler.

result Indicates the result of the simulation. 1 = passes ordinal adequacy test, 0 = fails ordinal adequacy test, OES = outside explanatory scope (in other words, this is not a result the model was designed to accommodate), 'pending' = the function listed in 'sim' is currently being written or tested.

sim The name of the catlearn function used to run the simulation.

oat The name of the catlearn function used to perform the Ordinal Adequacy Test.

Details

The Grid is a means of centrally recording the results of model simulations centrally, within the catlearn package. For further discussion, see Wills et al. (2016).

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

`citation('catlearn')`

References

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

Index

*Topic **datasets**

nosof88, [4](#)
nosof94, [13](#)
shin92, [22](#)
thegrid, [52](#)

*Topic **package**

catlearn-package, [2](#)

act2probrat, [3](#)

catlearn (catlearn-package), [2](#)
catlearn-package, [2](#)

nosof88, [4](#), [6–11](#), [13](#)
nosof88exalcove, [6](#), [7](#)
nosof88exalcove_opt, [6](#), [7](#)
nosof88oat, [6](#), [7](#), [8](#), [10](#), [13](#)
nosof88protoalcove, [9](#), [10](#), [11](#)
nosof88protoalcove_opt, [9](#), [10](#)
nosof88train, [5–7](#), [9](#), [10](#), [11](#)
nosof94, [13](#), [15–19](#)
nosof94bncalcove, [15](#), [16](#), [17](#)
nosof94exalcove, [16](#), [17](#), [18](#)
nosof94exalcove_opt, [16](#), [17](#)
nosof94oat, [14](#), [16](#), [17](#), [18](#), [21](#)
nosof94plot, [19](#)
nosof94train, [14–17](#), [20](#), [21](#)

optim, [7](#), [11](#), [18](#), [25](#), [28](#)

shin92, [22](#), [23–28](#), [30](#)
shin92exalcove, [23](#), [24](#), [25](#)
shin92exalcove_opt, [23](#), [24](#), [27](#)
shin92oat, [23](#), [25](#), [30](#)
shin92protoalcove, [26](#), [28](#)
shin92protoalcove_opt, [28](#)
shin92train, [22](#), [23](#), [26](#), [27](#), [29](#)
slpALCOVE, [6–11](#), [13](#), [15–18](#), [23–28](#), [30](#), [31](#)
slpBM, [33](#)
slpCOVIS, [36](#)
slpDIVA, [40](#)

slpMBMF, [42](#)
slpRW, [45](#)
ssecl, [47](#)
stsimGCM, [48](#)
thegrid, [52](#)