PSI.DLL Manual

Kai Schreiber

July 14, 2010

Contents

1 Overview **1**
1.1 Theory . . . 2
1.2 Implementation . . . 4
1.3 Files . . . 5

2 Exported functions **6**
2.1 SetupDist . . . 6
2.2 NextTrial . . . 7
2.3 StoreResult . . . 8
2.4 RemoveResult . . . 8
2.5 CurrentEstimate . . . 8
2.6 RescaleDist . . . 9
2.7 FinishDist . . . 11
2.8 DistActive . . . 11
2.9 HighestActive . . . 11
2.10 PossibleValues . . . 11
2.11 SavePFile . . . 12
2.12 SaveEFile . . . 12
2.13 SetDebugLevel . . . 12
2.14 GetDebugLevel . . . 12
2.15 EstimateTime . . . 12

3 Matlab class `psiSC` **13**

1 Overview

The method described in this document is based on the Ψ-method developed by Kontsevich and Tyler [Kontsevich & Tyler, 1999]. Their approach is expanded to include a third parameter, p_{err}. `psi.dll` provides a fast implementation of the Ψ-method. Libraries are provided for embedding `psi.dll` in Delphi, Visual Basic, Matlab and C/C++ programs.

On a Pentium 4 PC running Windows XP, this implementation delivers a new stimulus intensity for 30 different stimulus intensities and a probability array of size 200x50, representing 200 different values for `shift` and 50 different values for `slope` in about 135 ms. 50 stimulus intensities, and array dimensions of 500x50 still compute in under 600 ms, making this method useable without compromise on any modern Windows PC.

1.1 Theory

To estimate the psychometric function describing an observer's performance for a given task, `psi.dll` provides three different parametric psychometric functions, which must be chosen at the time of setting up the probability distribution. The index psf for choosing the function used is zero based.

These are: a logistic function (for detection tasks) (psf=0)

$$\Psi_{n_c,p_{err},s,c}(x) = \frac{1}{n_c} + \frac{h}{1 + e^{\frac{-4\tan(s)\cdot(x-c)}{h}}} \tag{1}$$

with

$$h = 1 - \frac{1}{n_c} - p_{err} \tag{2}$$

an inverted Gaussian (discrimination) (psf=1)

$$\Psi_{n_c,p_{err},s,c}(x) = 1 - p_{err} - h \cdot e^{-\left(\frac{\tan(s)\cdot(x-c)}{h}\right)^2} \tag{3}$$

with

$$h = 1 - p_{err} - \frac{1}{n_c} \tag{4}$$

and a logistic function with symmetrical lapse rates (discrimination) (psf=2)

$$\Psi_{n_c,p_{err},s,c}(x) = p_{err} + (1 - 2 \cdot p_{err})\frac{1}{1 + e^{\frac{-4\tan(s)\cdot(x-c)}{1-2\cdot p_{err}}}} \tag{5}$$

The parameters in these functions are n_c, the number of alternative answers the subject can give, p_{err}, the probability that the subject will make an error, s and c are the slope and threshold. The slope parameter ranges from -90 to 90 degrees and represents the slope at the threshold for the logistic functions, and the slope at 50% rise for the inverted Gaussian (the sign of the slope is irrelevant in this case).

For simplicity we collect the latter three of these parameters into parameter λ and consider n_c to be of fixed size.

1.1.1 The Probability Distribution

We now consider the probability distribution for these psychometric functions $p_{n_c}(\lambda)$. From now on I will consider n_c to be constant and omit it from the equations.

We start with some prior distribution $p_0(\lambda)$ and collect a set of data at intensities x_i with results (success/failure) r_i.

The probability for this set of data given λ is then the product of t he probabilites for the individual data points.

$$p(x_i, r_i|\lambda) = \prod_{i,r=s} \Psi_\lambda(x) \prod_{i,r=f} (1 - \Psi_\lambda(x)) \tag{6}$$

By Bayes' theorem we obtain the probability for each psychometric function given the data:

$$p(\lambda|x_i, r_i) = \frac{p(x_i, r_i|\lambda) \cdot p(\lambda)}{p(x_i, r_i)} p_0(\lambda) \tag{7}$$

Since $p(x_i, r_i)$ is an unknown constant and the left hand side has to be a normalized probability distribution, we can write

$$p(\lambda|x_i, r_i) = \frac{p(x_i, r_i|\lambda) \cdot p(\lambda) \cdot p_0(\lambda)}{\sum_\lambda p(\lambda)} \tag{8}$$

This means we can update the probability distribution after acquisition of result r_i at intensity x_i by multiplying it with the probability for this datapoint and renormalizing it.

$$p_i(\lambda) = \begin{cases} \dfrac{p_{i-1}(\lambda) \cdot \Psi_\lambda(x_i)}{\sum\limits_\lambda p_i(\lambda)} & r_i = success \\ \dfrac{p_{i-1}(\lambda) \cdot (1 - \Psi_\lambda(x_i))}{\sum\limits_\lambda p_i(\lambda)} & r_i = failure \end{cases} \tag{9}$$

1.1.2 The Expected Outcome

At any given time now we have the current probability distribution $p(\lambda)$. If the next trial were placed at intensity x_i, the probability for the result $r_i = success$ is

$$p(r_i = s|x_i) = \sum_\lambda p_{i-1}(\lambda) \Psi_\lambda(x_i) \tag{10}$$

Likewise, the probability for result $r_i = failure$ is

$$p(r_i = s|x_i) = \sum_\lambda p_{i-1}(\lambda)(1 - \Psi_\lambda(x_i)) \tag{11}$$

For each of these two outcomes we can compute the future probability distribution function $p_i(\lambda|r_i)$ using equation 9.

We now have the probability of each outcome and the future probaility distribution for these outcomes.

1.1.3 The Entropy of the Distribution

For estimating the quality or usefulness of a probability distribution function for estimating the parameters λ we use the entropy E of the distribution:

$$E = -\sum_{\lambda} p(\lambda) \ln(p(\lambda)) \tag{12}$$

Lower values of E signify more information about λ being represented in the distribution.

We can now compute the expected entropy for intensity x_i using the probability distributions obtained in equations 10 and 11:

$$< E >= -p(r_i = s, x_i) \cdot E(r_i = s, x_i) - p(r_i = f, x_i) \cdot E(r_i = f, x_i) \tag{13}$$

The intensity x_i, for which the expected entropy after the next trial becomes minimal will be the optimal intensity for that trial.

1.2 Implementation

The functions described in this manual have been written in Borland Delphi 7 for Windows and are part of the shared library psi.dll.

Using the header files included in the distirbution, the functions exported by `psi.dll` can be used in any programming language that supports the calling of such functions. Specifically, the functionality has been tested with Delphi, Visual Basic and Matlab 6.5(R13). A header file specifying the interface for use with C/C++ is also included as part of the interface to Matlab, but has not been used in C/C++ projects yet.

`psi.dll` exports the 15 funtions which are listed in this manual, using the `cdecl` calling convention. Sinve Visual Basic is unable to call external functions that use `cdecl`, I have included function wrappers using the `stdcall` calling convention. These functions have an addition `SC` in front of the name, i.e. the Visual Basic version of `SetupDist` is `SCSetupDist`.

1.2.1 The Mathwork's Matlab

To use `psi.dll` in Matlab, the Mathworks GenericDLL is required. This will only work in Matlab 6.5(R13). Once the GenericDLL package is installed, the collection of m-files included with `psi.dll` will make all the functions of the dll available from the Matlab command line.

Internally, each of the function wrapper m files checks whether the dll has been loaded and loads it if necessary. Then it calls the apropriate exported

dll function and returns the results. Most functions for matlab work identical to the below documentation, except for `CurrentEstimate`, which can be called omitting the parameter `e` for convenience, and will then return a 7x1 matrix containing all seven results at once.

1.2.2 Microsoft Visual Basic

For Visual Basic, the file `psi.bas` needs to be added to your program as a module. Make sure the dll file is present in your system's path. You can then call the functions from anywhere in your program.

1.2.3 Borland Delphi

To use `psi.dll` in Delphi just include the unit `psi.pas` in your project and include it in your uses clause. Put the dll in your system's path or the project output folder. You can now call all the dll functions.

1.2.4 C and C++

The file `psi.h` included for the Matlab interface contains the declarations to all the dll functions. This information should be usable to include the dll in C and C++ projects, but since I am not a C programmer I have not been able to write the full interface.

1.2.5 Other languages

`psi.dll` is usable in any language that can call external dlls using either the `cdecl` or `stdcall` calling conventions. Just load the dll using code apropriate for your language and declare the functions using the `.h`, `.pas` and `.bas` file as a reference.

1.3 Files

The distribution contains the following files

psi.dll The main library file

psi.h C header file declaring the functions

psi.bas Visual Basic Module declaring the exported functions

psi.pas Delphi unit declaring the exported functions

content.m Tabler of contents for Matlab's help system.

CurrentEstimate.m Matlab wrapper for psi.dll's function.

DistActive.m Matlab wrapper for psi.dll's function.

EstimateTime.m Matlab wrapper for psi.dll's function.

FinishDist.m Matlab wrapper for psi.dll's function.

GetDebugLevel.m Matlab wrapper for psi.dll's function.

HighestActive.m Matlab wrapper for psi.dll's function.

NextTrial.m Matlab wrapper for psi.dll's function.

PossibleValues.m Matlab wrapper for psi.dll's function.

RemoveResult.m Matlab wrapper for psi.dll's function.

RescaleDist.m Matlab wrapper for psi.dll's function.

SetDebugLevel.m Matlab wrapper for psi.dll's function.

SetupDist.m Matlab wrapper for psi.dll's function.

StoreResult.m Matlab wrapper for psi.dll's function.

UnloadPsi.m Removes psi.dll from Matlabs memory, if it has been loaded.

psiman.pdf This document.

psitest.exe A Windows executable for testing the functionality of psi.dll.

2 Exported functions

2.1 SetupDist

Syntax: `nd = SetupDist(nfc, nx, xmin, xmax, shift, nshift, sdshift, nsdshift, slope, nslope, sdslope, nsdslope, perr, nperr, sdperr, nsdperr, psf)`

Set up a new Gaussian or flat probability distribution prior and return its number. The parameters:

nfc This parameter represents the false alarm rate. If `nfc` is bigger than 1, it is interpreted as the number of choices and the false alarm rate is set to $\frac{1}{\texttt{nfc}}$. If `nfc` is smaller than 1, it is used directly as the false alarm rate. Pass 0 for a yes/no-type of experiment.

nx The number of distinct stimulus intensities.

nmin,nmax The minimum and maximum stimulus intensities, respectively.

shift The shift of the psychometric function. This is the intensity at which the function is halfway between guessing probability and $1 - p_err$

nshift The number of possible values for the shift to consider

sdshift The standard deviation of the gaussian prior for the shift

nsdshift The width in standard deviations around the shift that is included in the analysis

slope,nslope,sdslope,nsdslope The same four parameter types for the slope

perr,nperr,sdperr,nsdperr The same four parameter types for the miss rate

psf A parameter determining the psychometric function to use (see introduction)

In other words, `SetupDistErr` will setup a three dimensional probability distribution $p(\lambda)$, with λ centered on `shift, slope, perr`. The array containing the distribution will have `nshift x nslope x nperr` components, and will range from `(shift - nsdshift * sdshift)` to `(shift + nsdshift * sdshift)` for the `shift` dimension, from `(slope - nsdslope * sdslope))` to `(slope + nsdslope * sdslope)` in the `slope` dimension and from `(perr - nsdperr * sdperr)` to `(perr + nsdperr * sdperr)` in the `perr` dimension. If the lower bound for the `perr` dimension is below zero, the range will be truncated at zero[1].

To exclude a dimension from the analysis, set `nshift, nslope` or `nperr` to 1.

To make the prior for a dimension flat rather than a Gaussian, set `sdshift, sdslope` or `sdperr` to 0. The range of values considered in that dimension is then determined by the `nsd` parameter, which in this case becomes the absolute width.

`SetupDist` returns the number of the new distribution if succesful, -1 otherwise.

If you don't want to estimate the lapse rate, set `nperr` to 1. In this case, `sdperr` and `nsdperr` will be ignored.

`SetupDist` returns the number of the new distribution if succesful, -1 otherwise.

2.2 NextTrial

Syntax: `x = NextTrial(nd)`

This function returns the optimal stimulus intensity for placing the next trial for the probability distribution number `nd`. The intensity will be picked from the range of values specified using `(nx, xmin, xmax)` upon creation of the distribution.

This function will save a file `e.psi` containing the expected entropy function, if `DebugLevel` is set to 2 or higher.

If `nd` is a negative number, the actual `nd` used is $-1 - nd$. In this case the function does not return the stimulus intensity, but the index into the array of intensities, starting at 0 for `xmin` up to `nx-1` for `xmax`.

[1] i.e. a distribution with `perr=0.01, sdperr=0.02` and `nsdperr=2` will range from 0 to 0.05 and have a gaussian prior centered on 0.01. Because of the asymmetry introduced by the truncation the intital estimate for this prior will not be 0.01, however.

If the relevant distribution is not active, `NextTrial` returns -9999. The Matlab version of the function returns NaN in this case.

2.3 StoreResult

Syntax: `b = StoreResult(nd,x,r)`

This function stores a single trial and its outcome in the probability distribution `nd`, updating that distribution.

`x` is the intensity at which the trial was placed, `r` is the result, with `r=0` meaning failure and `r=1` meaning success.

The function returns 1 if the operation succeeded, 0 otherwise. Failure will be due to the specified distribution being inactive.

`StoreResult` will save the new probability distribution in a file named `p.psi`, if the `DebugLevel` is set to 2 or higher.

2.4 RemoveResult

Syntax: `b = RemoveResult(nd,x,r)`

Removes a stored result from a probability distribution. The syntax for specifying the result is the same as for `StoreResult`.

The function returns 1 if it was successful, 0 if it failed. Failure can be due to the specified distribution being inactive or the specified result not being stored in the distribution.

`RemoveResult` will save the new probability distribution in a file named `p.psi`, if the `DebugLevel` is set to 2 or higher.

2.5 CurrentEstimate

Syntax: `c = CurrentEstimate(nd,e)`

This function returns the expected values for the three components of λ and their standard deviations, the number of trials stored in a particular distribution, and the psf used to set it up.

$$< \lambda >= \sum_{\lambda} \lambda p(\lambda) \tag{14}$$

$$\begin{aligned} \sigma_\lambda &= \sqrt{< \lambda^2 > - < \lambda >^2} && (15) \\ &= \sqrt{\sum_{\lambda} \lambda^2 p(\lambda) - \left(\sum_{\lambda} \lambda p(\lambda) \right)^2} && (16) \end{aligned}$$

Each call to `CurrentEstimate` returns one component, based on the parameter `e`. For `e` ranging from 1 to 8 `CurrentEstimate` returns $< shift >$,

$< slope >$, $< perr >$, $< \sigma_{shift} >$, $< \sigma_{slope} >$, $< \sigma_{perr} >$, $ntrials$ and psf, respectively.

Passing negative `e` returns the unbiased estimates, i.e. removes the prior before computing the expected values.

`CurrentEstimate` returns -9999 if distribution `nd` is not active. The Matlab version of the function returns NaN in this case.

2.6 RescaleDist

Syntax: b = `RescaleDist(nd, method, nsdshift, nsdslope, nsdperr, nshift, nslope, nperr)`

This function is capable of changing the prior for a distribution, changing the scaling of a distribution and changing the resolution of a distribution, depending on the value of parameter `method`. The function returns 1 if the rescaling was successful, 0 otherwise.

`RescaleDist` will save the new probability distribution in a file named `p.psi`, if the `DebugLevel` is set to 2 or higher.

There are three main uses for this function:

1. When starting with a wide range of possible values for λ, the first few trials will restrict the probability distribution to a much narrower range. Rescaling to this new range will then increase the resolution of estimation, while leaving the computation time unchanged.

2. While a prior is useful in the beginning of an experiment to guide the placement of the first trials, it might actually introduce a bias later on. `RescaleDist` therefore can replace the prior with a flat distribution to remove that bias.

3. When the experiment is finished, it will be desirable to compute the expected values for the components of λ. This computation will be more accurate, the more elements per dimension the underlying probability distribution has. While the size of the distribution during an experiment is limited by intertrial intervals (see section /reffun:EsTi for reference), this limitation is grealty relaxed for analysis.

Any of the rescaling methods preserves all the data collected so far, by recreating a prior function and recalculating the current probability distribution from the stored trials. While changing resolutions, priors and scaling will affect the placement of future trials, it does not affect the viability of past or future data points.

The values of parameters not used by a particular method will be ignored.

2.6.1 `method=1`

The center and standard deviation of the prior will be (re)set to what it was when the distribution was first initialized. The new array of $p(\lambda)$ will be centered

on the current estimate of λ and the range of values will be set by the current estimate of σ_λ and the `nsd` parameters.

For example, the range of values for `shift` will be from `(<shift>- nsdshift*`σ_{shift}`)` to `(<shift> + nsdshift*`σ_{shift}`)`.

The number of elements of the array along each dimension will not be changed from what it is.

2.6.2 `method=2`

Same as method=1, but the prior is set to a flat distribution in all dimensions.

2.6.3 `method=3`

This resets the prior and the range of values for all three dimensions to what they were set to be when the distribution was first initialized.

2.6.4 `method=4`

This resets the range of values for all three dimensions to what they were set to be when the distribution was first initialized, but uses a flat prior.

2.6.5 `method=5`

Like `method=1`, but the number of elements along each dimension is reset to `nshift, nslope` and `nperr`.

2.6.6 `method=6`

Like `method=2`, but the number of elements along each dimension is reset to `nshift, nslope` and `nperr`.

2.6.7 `method=7`

Like `method=3`, but the number of elements along each dimension is reset to `nshift, nslope` and `nperr`.

2.6.8 `method=8`

Like `method=4`, but the number of elements along each dimension is reset to `nshift, nslope` and `nperr`.

2.6.9 Overview

Method	Rescale	Prior	Resize	Parameters used
1	yes	original	no	`nsdshift,nsdslope,nsdperr`
2	yes	flat	no	`nsdshift,nsdslope,nsdperr`
3	no	original	no	all ignored
4	no	flat	no	all ignored
5	yes	original	yes	all used
6	yes	flat	yes	all used
7	no	original	yes	`nshift, nslope, nperr`
8	no	flat	yes	`nshift, nslope, nperr`

2.7 FinishDist

Syntax: `b = FinishDist(nd)`

This will end data collection for probability distribution number nd. If `DebugLevel` is not 0, a file `distXXXX.psi` will be written containing the trial intensities and results. `XXXX` will be replaced with the number of the distribution used. Existing files will be overwritten. The distribution will be set to inactive and will no longer be accessible for any functions.

2.8 DistActive

Syntax: `b = DistActive(nd)`

Returns 1 if distribution `nd` is active, 0 otherwise.

2.9 HighestActive

Syntax: `h = HighestActive`

Returns the highest active distribution number. Returns -1 if no distribution is active.

2.10 PossibleValues

Syntax: `x = PossibleValues(nd,i)`

Returns the intensity value x_i for distribution `nd`. If $i = -1$, the function returns n_x for distribution `nd`.

This is useful if you have to precompute the stimuli for each intensity and need to determine the index of the intensity returned by `NextTrial`.

2.11 SavePFile

Syntax: `SavePFile(nd)`

Saves the probaility distribution number `nd` in the file `p.psi`, irrespective of the value of `DebugLevel`. In addition to `p.psi` the three scaling files `p_shift.psi, p_slope.psi` and `p_p_err.psi` will be saved. These contain the values of the parameters for each of the elements of the array stored in `p.psi`.

If p is a full three dimensional distribution array, the file `p.psi` contains a sequence of 2D arrays with perr constant for each array. The result is a `nshift x (nslope*nperr)` array.

2.12 SaveEFile

Syntax: `SaveEFile`

Saves the last computed expected entropy function, irrespective of the value of `DebugLevel`. The function saved will reflect the last call to `NextTrial` independent of the distribution used.

2.13 SetDebugLevel

Syntax: `SetDebugLevel(dl)`

Sets the internal `DebugLevel` flag. `dl=0` creates no file output, `dl=1` saves a file `trials.psi` whenever a distribution is inactivated using `FinishDistribution`, `dl=2` saves a file `p.psi` containing the probability distribution every time `StoreResult` or `RemoveResult` are called and a file `e.psi` containing the expected entropy function after each call to `NextTrial`.

2.14 GetDebugLevel

Syntax: `dl = GetDebugLevel`

Returns the current value of `DebugLevel`.

2.15 EstimateTime

Syntax: `t = EstimateTime(rep,nx,nshift,nslope,nperr)`

Use this function to measure the amount of time it takes `NextTrial` to compute the next trial's intensity on your machine. `rep` is the number of repetitions the measurement will be averaged over to improve accuracy, `nx` is the number of distinct trial intensities your experiment will contain, and `nshift, nslope` and `nperr` specifiy the size of the array used for estimation.

You should fix the number of stimulus intensities you want to use and then set `nshift`, `nslope` and `nperr` as high as possible without creating too much delay between trials.

The first time you call this function, it calibrates your processor's cycle counter against the PC clock to determine your machines frequency. This takes half a second. This does not influence the accuracy of the estimate for this first run.

3 Matlab class `psiSC`

For ease of handling, a matlab class defintion file is included in the distribution that serves as a wrapper for the specifc functions. The class object defined can be used to store trial results, retrieve the next trials intensity and obtain current estimates.

The class requires `psi.h` and `psi.dll` to be present in the class directory (`@psiSC`) to work.

Type `help psiSC` for a description of the class methods.

References

[Kontsevich & Tyler, 1999] Kontsevich, LL, & Tyler, CW. 1999. Bayesian adaptive estimation of psychometric slope and threshold. *Vis Res*, **39**(16), 2729–37.