(1) Overview

Title
QuestPlus: a MATLAB implementation of the QUEST+ adaptive psychometric method

Paper Authors
1. Jones, Pete Richard

Paper Author Roles and Affiliations
UCL Institute of Ophthalmology
11-43 Bath St
Greater London EC1V 9EL

Abstract
QuestPlus is a MATLAB implementation of the QUEST+ adaptive psychometric method. It provides a rapid and flexible method of estimating the parameters of a psychophysical model, and is also capable of advising the user on the most appropriate stimuli to present, and on when to terminate testing. Of particular note is the algorithm's ability to use prior information, its ability to determine the maximally informative stimulus on each trial, its ability to fit arbitrarily complex models, and its ability to vary multiple stimulus properties simultaneously.

Keywords
QUEST+, psychophysics, adaptive algorithms, Bayes, entropy, maximum likelihood, curve fitting, MATLAB

Funding statement:
This work was supported by the NIHR Biomedical Research Centre located at (both) Moorfields Eye Hospital and the UCL Institute of Ophthalmology.

Introduction
Psychophysics is a branch of experimental psychology concerned, primarily, with quantifying the limits of perception[1,2]. For example, we may wish to know what the faintest sound is that somebody can hear[3], or the finest spatial detail they can see[4]? This can be important in academic research (e.g., when studying the function of sensory systems), and also in clinical practice (e.g., when attempting to diagnose or monitor disease).

Adaptive psychophysical techniques attempt to answer such questions by systematically varying the parameters of a stimulus (e.g., the intensity of a sound), and finding the model that best fits the observer's responses (e.g., whether or not the observer successfully detected each stimulus).

In the case of QUEST+, the user specifies the parametric form of a model, and a set of possible values for each parameter ('hypotheses'). The primary role of QUEST+ is

to compute the posterior probability of each parameter value being true, given vectors of trial-by-trial stimulus values, *x*, and observed responses, *r*.

A detailed exposition of how QUEST+ does this is given elsewhere[5] (see also Refs~[6,7]). However, essentially it is operates using Bayes' Theorem. Thus, after *n* trials the posterior probability for a given set of parameters, θ, is given by:

$$p(\theta|\{x,r\}) = p(\theta)\prod_{i=1}^{n} p(r_i|\{x_i,\theta\}) \qquad \text{(Eq 1)}$$

where $\boldsymbol{p}(\theta)$ is the prior probability of each parameter value being true (NB: a uniform value, if no prior data is available), and x_i and r_i are pairs of stimulus values and observed responses, respectively, for each of the *n* trials.

Each update of the posterior requires the model to consider the likelihood of every possible response observation given every possible parameter combination and every possible combination of stimulus values. Under typical use the posterior distribution is updated after every observation. This presents a considerable computational challenge. In practice, the current implementation of QUEST+ minimizes the computational burden by precomputing all of these conditional probabilities in advance, and storing the results in a look-up table. However, this approach is still limited by the amount of available memory. For example, consider a task containing: two outcomes (e.g., 'correct' or 'incorrect'), a stimulus with two parameters (e.g., contrast and spatial frequency), each of which can take 50 possible values, and a model with five parameters, each of which can take 150 possible values. This implies 50^2 x 150^5 x 2 probabilities: more values than the largest possible double array that MATLAB supports (2^48-1 on 64-bit platforms). Typically, therefore, QUEST+ is applied to situations where the model contains no more than 4 parameters, and the stimulus has only one or two dimensions.

In addition to its core function of estimating parameter values, QUEST+ also has two secondary functions. First, QUEST+ is able to advise the experimenter on what stimulus value(s) to present next. QUEST+ does this by computing the stimulus that minimizes the *expected* negative Shannon entropy of the N-dimensional posterior probability density (i.e., of the N parameter estimates). Essentially, this can be thought of as the most *informative* stimulus. Note the experimenter is free to ignore the suggestions of QUEST+. For example, the experimenter may sometimes wish to present a very 'easy' stimulus to motivate the observer, rather than presenting the most informative stimulus on every trial.

Secondly, QUEST+ is also capable of advising the experimenter on when to stop testing (the Stopping Criterion). This is again computed based on the notion of entropy, and the precise mathematics can be found elsewhere in Ref~[5]. However, to give an intuition it should suffice to note that entropy is essentially a measure of how well a single hypothesis fits the evidence (i.e., a low value of negative Shannon entropy means that a considerable mass of the probability distribution is confined to

a set of small measure). Furthermore, in the simple case of a 1D posterior distribution, the entropy of the distribution is approximately proportional to its variance. Given that our goal is to have an infinitely narrow distribution, centered on the "true" parameter value, a minimum level of negative entropy provides a straightforward and principled cutoff point for testing. By default this cutoff is set at 3.0, but a serviceable value for a particular experiment can typically be found by trial-and-error. Moreover, as with stimulus placement, the experimenter is free to override the minimum entropy criterion, and determine their own Stopping Criterion. For example, users may wish to use a fixed number of trials, or to terminate the test early if the participant is becoming uncooperative.

Example application

Imagine one wanted to know how capable a particular individual was of telling apart the frequency (pitch) of two tones. We know, from previous research, that performance (proportion of trials correct: $\boldsymbol{p}(\boldsymbol{C})$) on a two-alternative frequency discrimination task is generally described by a 4-parameter cumulative Guassian function, shown graphically in Figure 1, and described mathematically as:

$$p(C) = \gamma + (1 - \lambda - \gamma)\Phi(x; \mu, \sigma) \quad \text{(Eq 2)}$$

[*Note to copyeditor: LaTeX sourcecode for Equations available*]

where $\boldsymbol{\Phi}$ is the cumulative Gaussian function, *x* is stimulus magnitude, and $\mu, \boldsymbol{\sigma}, \boldsymbol{\gamma}$, and $\boldsymbol{\lambda}$, are the four model parameters. The parameter $\boldsymbol{\gamma}$ represents the chance ('guess') rate, which, due to the nature of the task (two alternatives), is fixed at 0.5 (50%). The parameter $\boldsymbol{\lambda}$ represents the lapse rate (incorrect responses to seen stimuli, due to inattention or response errors), which we shall fix for now at a broadly representative value of 0.02 (2%). The parameters μ and $\boldsymbol{\sigma}$ relate to the sensitivity of the observer. We shall also fix $\boldsymbol{\sigma}$ for now at a representative value of 1 Hz, while μ is a free parameter that we wish to estimate. Based on prior data and piloting, we shall assume that the range of possible μ values is 1 – 20, and we shall posit 30 possible values ('hypotheses'), uniformly linearly-distributed within this range, as shown in Figure 1.

Finally, after specifying the model we must also specify the domain of possible responses (0: incorrect; 1: correct) and the domain of stimulus values, which in this case is a univariate parameter corresponding to the magnitude of the frequency difference between the two tones, in Hz. Given the model we are attempting to fit and the assumed range of possible parameter values, 40 possible stimulus values, log-distributed between 0.1 Hz and 100 Hz should be sufficient to constrain the model fit. Note, however, that these values should be refined through piloting and/or simulation, prior to running the final experiment.

Programmatically, the desired initialization is written as follows:

```
% set model
F = @(x,mu,sigma,gamma,lambda)gamma+(1 - gamma - lambda).*normcdf(...
    x,mu,sigma);

% set parameter domain(s)
mu       = linspace(1, 20, 30); % Sensitivity/threshold (free param)
sigma    = 1;                   % Sensitivity/slope (fixed param)
gamma    = 0.5;                 % Guess rate (fixed param)
lambda   = 0.02;                % Lapse rate (fixed param)
paramDomain = {mu, sigma, gamma, lambda};

% set stimulus domain(s)
stimDomain = logspace(log10(0.1), log10(100), 40);

% set response domain(s)
respDomain = [0 1];

% set stopping rule and criterion (or leave empty for default)
stopRule        = 'entropy';
stopCriterion   = 2;

% create QUEST+ object
QP = QuestPlus(F, stimDomain, paramDomain, respDomain, stopRule, ...
    stopCriterion);

% initialise (with default, uniform, prior)
QP.initialise();
```

[*Note to copyeditor: LaTeX sourcecode & Pdfs for Listings available*]

Then, during the test, we can query QUEST+ for suggested stimulus values, and update the QUEST+ with pairs of stimulus-response values (i.e., the actual stimulus value present, and the observed response). Thus:

```
while ~QP.isFinished()
    % get recommended stimulus
    targ = QP.getTargetStim();

    % ...
    % present stimulus and get response (anscorrect)
    % ...

    % update QUEST+
    QP.update(targ, anscorrect);
end
```

Finally, once the test is complete, the QUEST+ can be queried to provide parameter estimates given the current state of the posterior density function (Note, however, that when reporting data, users may wish to obtain more accurate estimates by refitting the model post-hoc):

```
% get final parameter estimates
endGuess_mean    = QP.getParamEsts('mean'); % ALT: QP.getParamEsts('mode')
est_mu           = endGuess_mean(1); % free parameter
est_sigma        = endGuess_mean(2); % fixed at: 1
est_gamma        = endGuess_mean(3); % fixed at: 0.5
est_lambda       = endGuess_mean(4); % fixed at: 0.02
```

A graphical illustration of this code in action is given in Figure 1, which shows how we initially posit 30 'hypotheses', which are then progressive ruled-out by successive empirical observations.

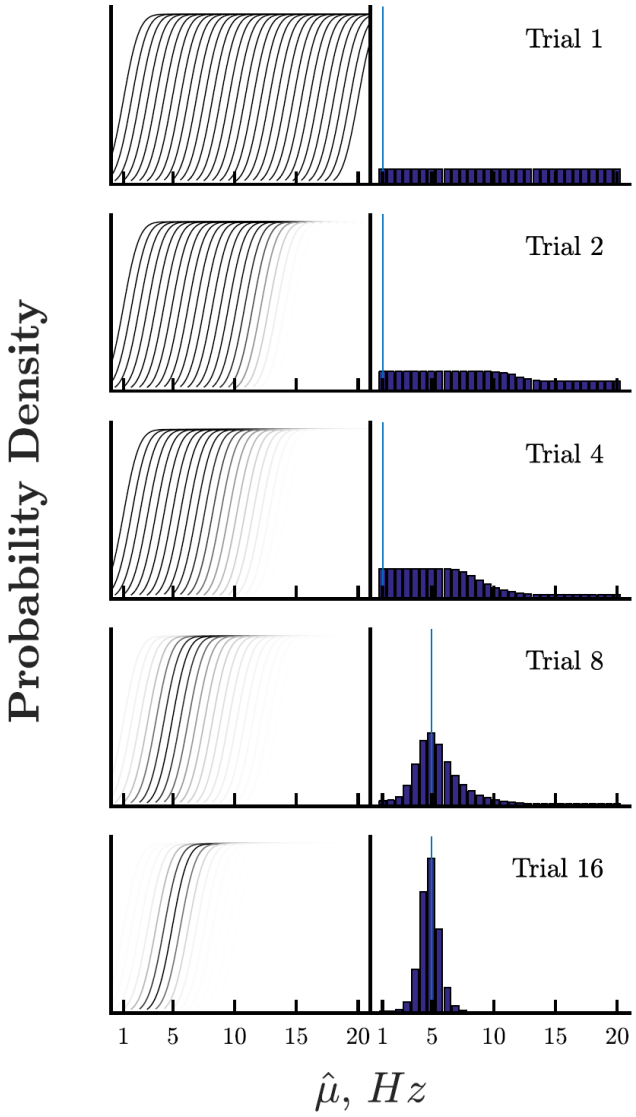

Figure 1. Graphical illustration of the example application, at the start of the first, second, fourth, eighth and sixteenth trial. Left panels show the 30 'hypotheses' (i.e., possible values of μ, given the model in Equation 2). The darkness of the curves is proportional to the relative likelihood of the estimate being true. Right panels show the posterior density function, with a blue vertical line denoting the current estimate (here based on the mean value). The stimuli, *x*, and observer's responses, *r*, aren't shown. However, it can be inferred that in the first few trials the observer successfully discriminated a number of mid-magnitude stimuli, making high values of μ (poor sensitivity) unlikely.

Finally, it is important to note that QUEST+ is highly flexible/generalizable. For example, it is trivial to: (i) vary the underlying model; (ii) make more than one value a free parameter; and (iii) add prior estimates to each of the parameter values, thus:

```
% set model
F = @(x,mu,sigma,gamma,lambda)gamma+(1 - gamma - lambda) .* ...
      cdf('wbl',10.^(x/20),10.^(mu/20),sigma);

% set true param(s)
mu = 5;
sigma = 1;
gamma = 0.5;
lambda = 0.02;
trueParams = {mu, sigma, gamma, lambda};

% set parameter domain(s)
mu      = linspace(1, 20, 30);
sigma   = linspace(1, 4, 30);
gamma   = 0.5;
lambda  = linspace(0.01, 0.1, 10);
paramDomain = {mu, sigma, gamma, lambda};

% set stimulus domain(s)
stimDomain = logspace(log10(0.1), log10(100), 40);

% set response domain(s)
respDomain = [0 1];

% create QUEST+ object
QP = QuestPlus(F, stimDomain, paramDomain, respDomain, [], 2.5);

% initialise with priors
y1 = normpdf(paramDomain{1},4,3);
y2 = normpdf(paramDomain{2},1,0.5);
y3 = 1;
y4 = normpdf(paramDomain{4},0.02,0.02);
% normalise so sum to 1
y1 = y1./sum(y1);
y2 = y2./sum(y2);
y3 = y3./sum(y3);
y4 = y4./sum(y4);
% set
priors = {y1, y2, y3, y4};
QP.initialise(priors)
```

For more examples of use, see Ref~[5], and see also the test cases embedded in in the current implementation (see *QuestPlus.runExamples*).

Implementation and architecture

The software is written entirely in MATLAB code, and all of the methods are contained within a single class: QuestPlus.m.

Quality control

End-to-end tests (which also serve as Minimal Working Examples) are incorporated within the QuestPlus class, and can be executed by running the static method QuestPlus.runExample(N), where N = 1, 2, …, 7. Test 6 in particular replicates an application from Ref~[5], who published numerical outputs that the present implementation has been validated against. QuestPlus is also being used for several studies within our lab.

(2) Availability

Operating system

QuestPlus is pure MATLAB code, and should function on all operating systems in which MATLAB is supported.

Programming language

QuestPlus requires MATLAB v2012 or later. . Due to the use of modern OOP syntax, QuestPlus is not compatible with earlier versions of MATLAB, or with Octave.

Additional system requirements

None.

Dependencies

There are no MATLAB dependencies in the core QUEST+ algorithm. However, several of the examples of use require the statistics toolbox.

List of contributors

Pete Jones wrote the software and is its current maintainer. Andrew B. Watson is the creator of the QUEST+ method.

Archive and Code Depository

GitHub

Name

QuestPlus

Persistent identifier

10.5281/zenodo.998564

URL

https://github.com/petejonze/QuestPlus

Licence

GNU GPL v3.0

Current version

v1.0.0

Publisher

Zenodo

Date published

28/09/2017

Language

English

(3) Reuse potential

QUEST+ is suitable for all psychophysical applications. Because of its extremely high efficiency, it is particularly well-suited to domains where the total number of trials is restricted, such as when working with children or patients. Furthermore, because of its complete flexibility, it is ideal for problems require the fitting of complex/arbitrary models. Currently, we are using it to estimate Contrast Sensitivity Functions in children with visual impairments.

Competing interests

The author has no competing interests to declare.

Acknowledgments

This work was supported by the NIHR Biomedical Research Centre located at (both) Moorfields Eye Hospital and the UCL Institute of Ophthalmology.

References

1. Gescheider, G. A.in 1997;73–124(Lawrence Erlbaum Associates, 1997).
2. Kingdom, F. A. A. & Prins, N. *Psychophysics: a practical introduction*. 2010;(Elsevier Academic Press, 2010).
3. Jones, P. R., Moore, D. R. & Amitay, S. Development of auditory selective attention: Why children struggle to hear in noisy environments. *Dev. Psychol.* 2015;51:353
4. Jones, P. R., Kalwarowsky, S., Atkinson, J., Braddick, O. J. & Nardini, M. Automated measurement of resolution acuity in infants using remote eye-tracking. *Invest. Ophthalmol. Vis. Sci.* 2014;55:8102–8110
5. Watson, A. B. QUEST+: A general multidimensional Bayesian adaptive psychometric methodWatson. *J. Vis.* 2017;17:10
6. Kontsevich, L. L. & Tyler, C. W. Bayesian adaptive estimation of psychometric slope and threshold. *Vision Res.* 1999;39:2729–2737
7. Watson, A. B. & Pelli, D. G. QUEST: A Bayesian adaptive psychometric method. *Percept. Psychophys.* 1983;33:113–120