The optseq run from the command window.

You can download the optseq from here:

<https://surfer.nmr.mgh.harvard.edu/optseq/>

First, cd to the directory where the optseq files are and the output will be saved

**Cd ...**

Second, activate the optseq – write in the terminal command line

**chmod a+x** **optseq2**

Now define the inputs and run the optseq to get the optimal sequence and jitter of ITI

* the mark “--” represent inserting a new input.
* ‘null’ trials- is the jitter, cause it’s duration is changes in the optimal way to increase efficiency. Determine it’s limits by ‘tnullmin’ and ‘tnullmax’

Ntp: number of time points in the run. Composed of mean duration of trials (including jitter, meaning the SOA) times number of trials in the run, divided by TR. The Total Scanning Time is the number of time points times the TR plus the prescan period,i.e., tScanTot = Ntp\*TR+tPreScan.

* Psdwin – the time window to catch response. I set it to the trial length. If it is too long it makes problems. the third number determines the steps of change in duration of the ‘null’ (jitter) events
* Ev – define each trial type: label, duration (in sec), number of repetitions. The duration should be the minimal duration of the trial, before the jitter.
* Nkeep – how many sequences to save. Can correspond with the number of runs times the number of orders to counterbalance
* --o: the name of result file
* The result sequence will contain the optimal order of the trials with jitter (named ‘null’) in between
* After adjusting the inputs to your experiment, copy the paragraph to the command window and run (press enter)
* The result file ‘.par’ can be opened in matlab
* -- mtx saves design matrix as mat file

**./****optseq2 --ntp 122 --tr 2 --tprescan 2 --psdwin 0 10 1 --ev Faces\_H 8 3 --ev Faces\_L 8 3 --ev Fractals\_H 8 3 --ev Fractals\_L 8 3 --ev Snacks\_H 8 3 --ev Snacks\_L 8 3 --ev Money\_H 8 3 --ev Money\_L 8 3 --nkeep 30 --o OptseqResults --tsearch 0.1 --tnullmin 0 --tnullmax 4**

**optseq2 –ntp131 --tr 2 --tprescan 2 --psdwin 1 10 2 --ev H \_val\_\_ H \_efrt 1.5 3 --ev H \_val\_\_ M \_efrt 1.5 3 --ev H \_val\_\_ L \_efrt 1.5 3 --ev M \_val\_\_ H \_efrt 1.5 3 --ev M \_val\_\_ M \_efrt 1.5 3 --ev M \_val\_\_ L \_efrt 1.5 3 --ev L \_val\_\_ H \_efrt 1.5 3 --ev L \_val\_\_ M \_efrt 1.5 3 --ev L \_val\_\_ L \_efrt 1.5 3 --nkeep 10 --o OptseqResults --tsearch 0.****1 --tnullmin 0 --tnullmax 4**

**Optseq manual**

USAGE: optseq2

Data Acquistion Parameters

--ntp Ntp : number of time points

--tr TR : temporal resolution of acquisition (in sec)

--tprescan t : start events t sec before first acquisition

Event Response and Nuisance Descriptors

--psdwin psdmin psdmax <dPSD> : PSD window specifications

--ev label duration nrepetitions

--repvar pct <per-evt>: allow nrepetitions to vary by +/- percent

--polyfit order

--tnullmin tnullmin : limit min null duration to tnullmin sec

--tnullmax tnullmax : limit max null duration to tnullmax sec

Searching and Cost Parameters

--nsearch n : search over n schedules

--tsearch t : search for t hours

--focb n : pre-optimize first order counter-balancing

--ar1 rho : optimize assuming whitening with AR1

--pen alpha T dtmin: penalize for presentations being too close

--evc c1 c2 ... cN : event contrast

--cost name <params>: eff, vrfavg, vrfavgstd

--sumdelays : sum delays when forming contrast matrix

--seed seedval : initialize random number generator to seedval

Output Options

--nkeep n : keep n schedules

--o outstem : save schedules in outstem-RRR.par

--mtx mtxstem : save design matrices in mtxstem\_RRR.mat

--cmtx cmtxfile : save contrast matrix in cmtxfile

--sum file : save summary in file (outstem.sum)

--log file : save log in file (outstem.log)

--pctupdate pct : print an update after each pct done

--sviter file : save info from each iteration in file

Input/Initialization Options

--i instem : initialize with instem-RRR.par

--in input-schedule <--in input-schedule >

--nosearch : just print output for input files

Help, Documentation, and Bug Reporting

--help : print help page

--version : print version string

$Id: optseq2.c,v 2.12 2006/12/29 02:09:11 nicks Exp $

Optseq Home Page:

http://surfer.nmr.mgh.harvard.edu/optseq

SUMMARY

optseq2 is a tool for automatically scheduling events for

rapid-presentation event-related (RPER) fMRI experiments (the schedule

is the order and timing of events). Events in RPER are presented

closely enough in time that their hemodynamic responses will

overlap. This requires that the onset times of the events be jittered

in order to remove the overlap from the estimate of the hemodynamic

response. RPER is highly resistant to habituation, expectation, and set

because the subject does not know when the next stimulus will appear

or which stimulus type it will be. RPER is also more efficient than

fixed-interval event related (FIER) because more stimuli can be

presented within a given scanning interval at the cost of assuming

that the overlap in the hemodynamic responses will be linear. In SPM

parlance, RPER is referred to as 'stochastic design'.

The flexibility of RPER means that there are a huge number of possible

schedules, and they are not equal. optseq2 randomly samples the space

of possible schedules and returns the 'best' one, where the user can

control the definition of 'best'. Cost functions include: average

efficiency, average variance reduction factor (VRF), and a weighted

combination of average and stddev of the VRF. The user can also

specify that the first order counter-balancing of the sequence of

event-types be pre-optimized.

Visit the Optseq Home Page at:

http://surfer.nmr.mgh.harvard.edu/optseq

COMMAND-LINE ARGUMENTS

--nt

Ntp

Number of time points to be acquired during the scan. This should be

for one 'run' not for the entire session. The Total Scanning Time

is the number of time points times the TR plus the prescan period,

ie, tScanTot = Ntp\*TR+tPreScan.

--tr TR

Time between functional volumes (in seconds).

--tprescan tPreScan

Time before the acquisition of the first volume to be processed to

begin stimulation.

--psdwin PSDMin PSDMax <dPSD>

Specifications for the FIR event response window. It will be assumed that

the entire response can be captured within this window. PSDMin is the

minimum PostStimulus Delay (PSD), PSDMax is the maximum PSD. dPSD

is the sampling interval within the window. dPSD is optional; if

left unset, it will default to the TR. dPSD controls how finely spaced

the event onsets can be scheduled (ie, the onsets will only appear at

integer multiples of the dPSD).

--ev label duration nrepetitions

Event Type specification. The label is just a text label (which may be

more informative than a numeric id). Duration is the number of seconds

that the stimulus will be presented; it should be an integer multiple

of the dPSD (see --psdwin). Nrepetitions is the number of times that

this event type will be presented during the course of the run. The

number of repetitions can be optimized using the --repvar option. Use

a different --ev flag for each event type. NOTE: DO NOT INCLUDE THE

NULL STIMULUS AS AN EVENT TYPE. The total stimulation time, tStimTot,

equals the product of the duration and the number of repetitions

summed over all the events. It should be obvious that the total

stimulation time must be less than the total scanning time.

--repvar pct <per-evt>

Allow the number of repetitions of each event type to randomly vary by

+/- pct percent from the number specified with --ev. This allows the

user to optimize over the number of repetitions. The total stimulation

time is computed from the maximum possible number of repetitions. If

only the percentage is given, then the relative number of repetitions

of each event type will stay constant. If the string 'per-evt' is

appended, then the number of reps for each event type can change

independently to each other.

--polyfit order

Add polynomial regressors as nuisance variables. Order N includes the

Nth order polynomial was well as all lower orders. Max order is currently

2. Order 0 is a baseline offset; Order 1 is a linear trend; Order 2

is a quadradic trend. Cost functions will not explicitly include the

nuisance variables.

--tnullmin tNullMin

Force the NULL stimulus to be at least tNullMin sec between stimuli.

Note that this means that the stimulus duration + tNullMin must be

an integer multiple of the dPSD.

--tnullmax tNullMax

Limit the maximum duration of the NULL stimulus to be tNullMax sec.

Note: it may not be possible for a given parameter set to keep the NULL

stimulus below a certain amount. In this case, the following error

message will be printed out 'ERROR: could not enforce tNullMax'. By

default, tNullMax is infinite.

--nsearch Nsearch

Search over Nsearch iterations. optseq will randomly construct Nsearch

schedules, compute the cost of each one, and keep the ones with the

highest cost. It is not permitted to specify both Nsearch and Tsearch.

--tsearch Tsearch

Search for Tsearch hours. optseq will randomly construct as many

schedules schedules as it can in the given time, compute the cost of

each one, and keep the ones with the highest cost. It is not

permitted to specify both Nsearch and Tsearch.

--focb nCB1Opt

Pre-optimize the first order counter-balancing (FOCB) of the event

sequence. This will cause optseq2 to construct nCB1Opt random

sequences and keep the one with the best FOCB properties. This will be

done for each iteration. Counter balance optimization is not allowed

when there is only one event type.

--ar1 rho

Optimize while whitening with an AR(1) model with parameter rho. rho must

be between -1 and +1.

--pen alpha T dtmin

Penalize for one presentation starting too soon after the previous

presentation. The weight is computed as 1 - alpha\*exp(-(dt+dtmin)/T), where

dt is the time from the offset of the previous stimulus to the onset

of the next stimulus. The basic idea here is that the second stimulus

will be reduced in amplitude by the weight factor. alpha and T were fit

from data presented in Huettel and McCarthy (NI, 2000) to be alpha=0.8

and T = 2.2 sec.

--evc C1 C2 ... CN

Optimize based on a contrast of the event types. Ci is the contrast

weight for event type i. There must be as many weights as event types.

Weights are NOT renormalized such that the sum to 1.

--cost costname <params>

Specify cost function. Legal values are eff, vrfavg,

vrfavgstd. Default is eff. params as any parameters which accompany

the given cost function. eff is the cost function which maximizes

efficiency (no parameters). vrfavg is the cost function which

maximizes the average Variance Reduction Factor (VRF) (no

parameters). vrfavgstd maximizes a weighted combination of the average

and stddev VRF; there is one parameter, the weight give to the stddev

component.

--sumdelays

Sum the delay regression parameters when computing contrast matrix.

The event contrast (--evc) specifies how to weight the events when

forming the contrast vector. However, there are multiple coefficients

per event type corresponding to the delay in the FIR window. By default,

a separate row in the contrast matrix is provided for each delay. To

sum across the delays instead, use --sumdelays. The contrast matrix

will have only one row in this case.

--seed seedval

Initialize the random number generator to seedval. If no seedval is

specified, then one will be picked based on the time of day. optseq2

uses drand48().

--pctupdate pct

Print an update line to stdout and the log file after completing each

pct percent of the search.

--nkeep nKeep

Save nKeep of the best schedules. Increasing this number does not

substantially increase the search time, so it is a good idea to

specify more than you think you will need.

--o outstem

Save schedules in outstem-RRR.par, where RRR is the 3-digit

zero-padded schedule rank number (there will be nKeep of them).

The schedules will be saved in the Paradigm File Format (see below).

--mtx mtxstem

Save the FIR design matrices to mtxstem\_RRR.mat in Matlab 4 binary

format.

--cmtx cmtxfile

Save the contrast matrix in Matlab 4 binary format.

--sum summaryfile

optseq2 will create a file which summarizes the search, including

all the input parameters as well as characteristics of each of

the schedules kept. By default, the summary file will be outstem.sum,

but it can be specified explicitly using this flag. See THE SUMMARY

FILE below.

--log logfile

During the course of the search, optseq2 will print information about

the current search status to stdio and to the log file. By default

the log file will be outstem.log. The log file will contain a summary

of input arguments as well as a series of status lines. A status line

will be printed each time there is a change in the list of nKeep best

schedules as well as at prespecified regular intervals. By default,

the interval is 10% of the search time, but this can be changed

with --pctupdate. Each status line has 12 columns: (1) percent complete,

(2) iteration number, (3) minutes since start, (4) best cost,

(5) efficiency, (6) CB1Error, (7) vrfavg, (8) vrfstd, (9) vrfmin,

(10) vrfmax, (11) vrfrange, and (12) number of iterations since

last substitution.

--pctupdate pct

Print a search status to stdio and the log file at regular intervals

corresponding to pct percent of the search time. Default is 10%.

--sviter SvIterFile

Save information summary about all the schedules to SvIterFile in

ASCII format. Each line will have 7 columns corresponding to:

(1) cost, (2) efficiency, (3) cb1err, (4) vrfavg, (5) vrfstd,

(6) vrfmin, (7) vrfmax. This is mainly for exploring the distribution

of the various costs. WARNING: this file can grow to be very large.

--i instem

Load all input schedules that match instem-RRR.par. These can be used

to initialize the search (for example, if you want to continue a

previous optimization). It is also possible to only generate a summary

and/or design matrices of the given input schedules by include the

--nosearch flag. This can be useful for testing schedules that were

optimized under one cost function against another cost function or

for testing independently generated schedules. See also --in.

--in input-schedule <--in input-schedule >

This does the same thing as --i except that each file is specified

separately.

--nosearch

Do not search for optimal schedules. This can only be used when

reading schedules in using --i or --in. See --i for more information.

ALGORITHM OVERVIEW

optseq2 randomly searches the space of schedules given the constraints

on the command-line and keeps the ones that maximize the given cost

function. Each search iteration begins by creating a random order of

events with the appropriate number of repetitions for each event

type. First order counter-balancing optimization, if done, is

performed here. Next, the timing is generated by inserting random

amounts of NULL stimulus so that the total stimulation time plus null

time is equal to the total scan time. Event onset times are

constrained to be integer multiples of dPSD. An FIR design matrix is

created from this schedule. The FIR peristimulus window begins at

PSDMin and ends at PSDMax and is incremented by dPSD. If polynomial

regressors are specified, they are appended to the FIR matrix to give

the final design matrix, hereafter referred as X. The various costs

are computed from X. The forward model is then y = XB+n, which has the

solution Bhat = inv(XtX)Xy. A contrast is Ghat = C\*Bhat, where C is the

the contrast matrix.

CONTRAST MATRIX

By default, the contrast matrix is the identity over all task-related

components. The contrast matrix can be changed by specifying --evc

(and possibly --sumdelays).

COST FUNCTIONS

First-Order Counter-Balancing (FOCB). The FOCB matrix is the

Nevt-by-Nevt matrix of probabilities that one event type follows

another, where Nevt is the number of event types (excluding the NULL

condition). This is computed only from the sequence of events and is

independent of the timing (this is why it is referred to as

'pre-optimization'). The ideal FOCB matrix can be computed from the

number of repetitions for each event type. The FOCB cost matrix is

computed by subtracting the actual probability from the ideal and then

dividing by the ideal. The final cost is computed by averaging the

absolute values of all elements in the cost matrix. This cost is

minimized during pre-optimization. FOCB optimization can be combined

with any other cost function. Note: FOCB requires that there be at

least 2 event types.

Efficiency (eff). Efficiency is defined as eff = 1/trace(C\*inv(Xt\*X)\*Ct)

(note: any nuisance regressors are not included in the computation of

the trace but are included in the computation of the inverse). The

quantity trace(C\*inv(XtX)\*Ct) is a measure of the sum square error in Ghat

(ie, G-Bhat) relative to the noise inherent in the experiment. Therefore,

maximizing eff is a way of finding a schedule that will result in, on

average, the least error in Ghat.

Average Variance Reduction Factor (vrfavg). The Variance Reduction Factor

(VRF) is the amount by which the variance of an individual estimator (ie,

a component of Ghat) is reduced relative to the noise inherent in the

experiment. The VRF for a estimator is the inverse of the corresponding

component on the diagonal of C\*inv(XtX)\*Ct. The average VRF is this value

averaged across all estimators. This will yield similar results as when

the efficiency is optimized.

Average/StdDev Variance Reduction Factor (vrfavgstd). The cost is defined

as cost = vrfavg - W\*vrfstd, where vrfstd is the standard deviation of

the VRFs and W is a weighting factor specified as a parameter on the

command-line. This penalizes schedules that result in large variations

in the individual VRFs of the estimators. There is currently a bug in

the implementation that causes it to mis-state the cost when the number

of repetitions are different for different event types. Also, only use

this cost when using a prescan window equal to or greater than the

PSD window (otherwise there will be a tendency not to schedule events

near the end of the run).

THE SUMMARY FILE

The summary file summarizes the conditions under which the search was

performed as well as the properties of each schedule found. It also

includes the number of iterations searched and the time it took to

search them as well as the average and standard deviation of the cost

measured over all schedules. It also includes the maximum efficiency

and average VRF over all schedules (these will be the same as the

best schedule if the eff or vrfavg cost functions were chosen).

Each schedule is summarized in a table with the following columns:

(1) Rank, (2) Cost, (3) ZCost, (4) Iteration Number (NthIter), (5)

Efficiency (Eff), (6) FOBC Error (CB1Err), (7) Average VRF (VRFAvg),

(7) StdDev VRF (VRFStd),

(8) Minimum VRF (VRFMin), (9) Maximum VRF (VRFMax), and (10) VRF

Range (VRFRng). Many of these measures have been described above.

ZCost is the number of standard deviations from the average cost

(over all schedules). The Iteration Number is the search iteration that

that schedule was found on. The first-order counter-balancing

measures come after this table. First, the ideal FOCB probability

matrix is printed followed by the actual matrix for each of the

schedules. Note: the printed ideal matrix is based on the nominal number

of repetitions. See BUGS.

CHOOSING PARAMETERS SETS

There are several parameters that must be chosen as a group because

they rely and/or relate to each other. These parameters are: (1) the

number of time points (Ntp), (2) the TR, (3) the prescan window (tPreScan),

(4) the duration of each event type (tEv), and (5) the number of repetitions

of each event type (nReps). The most basic relationship requires that

the total amount of stimulation time (tStimTot) be less than or equal to

the total amount of scan time (tScanTot), where tStimTot = sum(tEv\*nReps)

(summed over all conditions), and tScanTot = Ntp\*TR+tPreScan, so

sum(tEv\*nReps) <= Ntp\*TR+tPreScan (1)

If this constraint is not met, you will receive a 'Time Constraint

Violation' Error. The total amount of time dedicated to the Null stimulus

(tNullTot) is equal to the difference between the total scan time and

the total stimulation time:

tNullTot = Ntp\*TR+tPreScan - sum(tEv\*nReps) (2)

If the parameters are chosen such that equality results in equation (1),

then there will not be any time for the Null stimulus, which is generally

not good because inter-stimulus jitter is dependent upon inserting

random amounts of Null between non-Null stimuli.

A rule of thumb is to allocate as much time for the Null as one would for

any other stimulus. This can be done by choosing parameters such that

sum(tEv\*nReps)(nEv+1)/nEv = Ntp\*TR+tPreScan (3)

where nEv is the number of event types. The schedule can be optimized

around this point by allowing the number of repetitions to vary around

this nominal value.

There is also a DOF constraint which requires that the number of parameters

estimated be less than the number of time points, ie

Nbeta = nPSD\*nEv+(PolyOrder+1) < Ntp (4)

where Nbeta is the number of parameters, nPSD is the number of elements

in the post-stimulus time window (ie, (PSDMax-PSDMin)/dPSD), and PolyOrder

is the order of the nuisance polynomial specified with -polyfit. If this

constraint is not met, you will receive a 'DOF Constraint Violation' Error.

CHOOSING THE SEARCH TERMINATION CRITERIA

The search is terminated when either the maximum number of iterations

has been reached or the maximum search time has been reached. It is

impossible to determine how many iterations to search over because

it is not possible to globally determine what the best schedule is nor

would it be possible to determine how long it would take a random search

to get there. That said, there are some rules of thumb that can be followed,

the most basic being that if a 'large' number of schedules have been

searched and the best cost has not changed 'much' in a 'long time', then

you are done. Of course, you still have to define 'large', 'much', and

'long time'. The summary file can help with this. In particular, there is a

line with the number of iterations since the last substitution (ie,

the number of iterations since one of the best nKeep schedules changed).

This can be used to judge how long a 'long time' is. The same information

can be extracted from the NthIter column of the summary table. At a minimum

let it run for 10000 iterations.

PARADIGM FILE FORMAT

The schedules will be saved in 'paradigm file' format. This format has four

columns: time, numeric event id, event duration, and event label. A numeric

id of 0 indicates the Null Stimulus.

BUGS

Also see the Optseq Home page at http://surfer.nmr.mgh.harvard.edu/optseq

The vrfavgstd cost function does not work properly if the number of reps

is different for different event types. A prescan window should also be

specified

The ideal counter-balance matrix reported in the summary file will be

for the nominal number of reps when the user has selected to optimize

over the number of reps making comparisons between the actual and

ideal inappropriate (the FOCB error reported for each will be correct

however).

BUG REPORTING

optseq2 is offered with no guarantees and with no obligation or implication

that it will be supported in any way. Having said that, you can send

bug reports/questions to: analysis-bugs@nmr.mgh.harvard.edu. You must

include the following information: (1) optseq2 version, (2) computer

operating system, (3) command-line, (4) description of the problem, and

(5) log and/or summary files.

AUTHOR

optseq2 was written by Douglas N. Greve in the Summber of '02.

REFERENCES

Dale, A.M., Greve, D.N., and Burock, M.A. (1999) Optimal Stimulus

equences for Event-Related fMRI. 5th International Conference on

Functional Mapping of the Human Brain. Duesseldorf, Germany. June

11-16.

Dale, A.M. (1999). Optimal experimental design for event-related

fMRI. Human Brain Mapping 8:109-114.