Neurodesign: Optimal experimental designs for task fMRI

Joke Durnez[1,2,3], Ross Blair[1,2], Russell A. Poldrack[1,2]

[1]*Department of Psychology, Stanford University,*
[2]*Stanford Center for Reproducible Neuroscience, Stanford University*
[3]*Parietal, Neurospin, Institut National de Recherche en Informatique et en Automatique*

March 22, 2017

Abstract

A recent stream of alarming publications questions the validity of published neuroimaging findings. As a consequence, fMRI teams worldwide are encouraged to increase their sample sizes to reach higher power and thus increase the positive predictive value of their findings. However, an often-overlooked factor influencing power is the experimental design: by choosing the appropriate experimental design, the statistical power of a study can be increased within subjects. By optimizing the order and timing of the stimuli, power can be gained at no extra cost. To facilitate design optimization, we created a **python** package and web-based tool called Neurodesign to maximize the detection power or estimation efficiency within subjects, while controlling for psychological factors such as the predictability of the design. We implemented the genetic algorithm, introduced by Wager and Nichols (2003) and further improved by Kao et al. (2009), to optimize the experimental design. The toolbox allows more complex experimental setups than existing toolboxes, while the GUI provides a more user-friendly experience. The toolbox is accessible online at www.neuropowertools.org.

1 Introduction

A recent stream of alarming publications questions the validity of published neuroimaging findings (Eklund et al., 2016; Ioannidis, 2005; Open Science Collaboration, 2015). At the core of the reproducibility crisis is the lack of power typically observed in neuroimaging Button et al. (2013), and more specifically, fMRI studies (Durnez et al., 2014). The signal measured in fMRI is known to

be very noisy, while the hypothesised effects are small, such that a push for larger sample sizes promises a more powerful future for neuroimaging. Different power analysis strategies offer a way to optimise the sample size for a specific power level (Durnez et al., 2014; Mumford and Nichols, 2008; Hayasaka et al., 2007; Durnez et al., 2016). However, fMRI data are typically acquired and aggregated on two levels: within and between subjects. As such, increasing the power of an fMRI experiment can be achieved by increasing the number of subjects, but also via the within subjects experimental design. This is especially true for smaller and more subtle effects, where the power curve is characterised by a slower increase, and thus the resulting power is more affected by the number of subjects and the number of time points. In addition to the duration of the experiment for each subject, the order and timing of different conditions within the experiment also influence the power of the resulting analyses. The goal in task fMRI experiments is often one of two: detection or estimation. Detection refers to detecting the difference in brain activation between conditions or groups, while estimation relates to estimating the exact shape of the evoked fMRI response (called the haemodynamic response function, HRF). Ideally, the design of an fMRI experiment changes according to the specific research question asked. An optimal design with respect to these two distinct research questions are said to maximize the detection power or the estimation efficiency respectively. It is often argued that those two goals are opposite and an increase in detection power inevitably leads to a decrease of estimation efficiency. For example, when two trials of the same condition follow each other closely, the signal tends to accumulate linearly (Dale, 1999), which makes it easier to detect. Therefore, the experiments often consist of blocks of the same condition. This type of design is called a blocked design. On the contrary, the accumulation (and saturation) of the measured signal conceals the shape of the HRF. To estimate the HRF, scientists often opt for an event-related design, where both the timing and order of conditions are randomised. However, Kao et al. (2009) show that the necessary trade-off between detection and estimation can be improved using certain optimisation algorithms. Another important aspect in an fMRI design is the psychological experience of the subject in the scanner. With a blocked design, the design becomes very predictable for subjects which can potentially bias the psychological function hypothesised in the first place. To minimise the predictability, Buracas and Boynton (2002) propose the use of m-sequences. Very often, the best design is a combination of a maximal signal with low predictability. Therefore, Wager and Nichols (2003) suggest the use of a genetic algorithm to find an optimisation between estimation efficiency, detection power and predictability. This algorithm optimises a weighted average of different outcome measures, with the weights depending on the hypothesis and the expected outcome of the experiment. Later, the algorithm has been further fine tuned and compared with other approaches (Kao et al., 2009). In this paper, we present an implementation of the genetic algorithm for fMRI designs, which is both available as a python module as well as a GUI web tool, available at www.neuropowertools.org. The paper is structured as follows: we start with a general description of the methodology in section 2. We show how

designs can be compared and optimised using our python module in Section 3. An overview of the GUI is given in Section 4. Finally, we conclude and compare to other existing software in Section 5.

2 Design optimisation using the genetic algorithm

2.1 Statistical measures of design optimality

The signal measured using fMRI is the blood oxygen level dependent (BOLD) signal, which is a assumed to be related to the neural signal via convolution with a hemodynamic response function (HRF). We consider the general linear model as the underlying model for the statistical objective:

$$Y = X\beta + \epsilon, \epsilon \sim N(0, \sigma).$$

We denote Y as the measured signal. Z represents the design matrix, β is the response amplitude for each column/condition in X and ϵ the error. We consider two types of design matrices X: the convolved model and the finite impulse response (FIR) model. Figure 1 shows an example of both models. The first model aims to estimate the amplitude of the signal, while the goal of the latter is estimating the exact shape of the HRF.

Often, researchers are interested in specific hypotheses concerning particular combinations of parameters. The parameter of interest can be estimated using the least squares estimator:

$$\hat{\beta}_c = c'(X'X)^{-1}X'Y,$$

and its variance,

$$Var(\hat{\beta}_c) = \sigma^2 c(X'X)^{-1}c',$$

with c the contrast vector of interest. To account for the specific character of fMRI data, we alter the model slightly. Because fMRI timeseries data exhibit substantial temporal autocorrelation, the data are assumed to follow the covariance matrix V, where off-diagonal values represent the correlation between measurements at different time points. Furthermore, a regressor S, representing low-frequency noise components, is added to the model (see Kao et al. (2009) for detailed derivations). The resulting variance of the estimator becomes:

$$Var(\hat{\beta}_c) = \sigma^2 c(X'WX)^{-1}c',$$

with $W = V - VS'(SVS')^{-1}SV$. An optimal experimental design with respect to the estimator minimises the variance of the estimator. We will therefore quantify the optimality of the design as $c(X'WX)^{-1}c'$. Most often, an fMRI experiment has multiple contrasts of interest, $c(X'WX)^{-1}c'$ becomes a

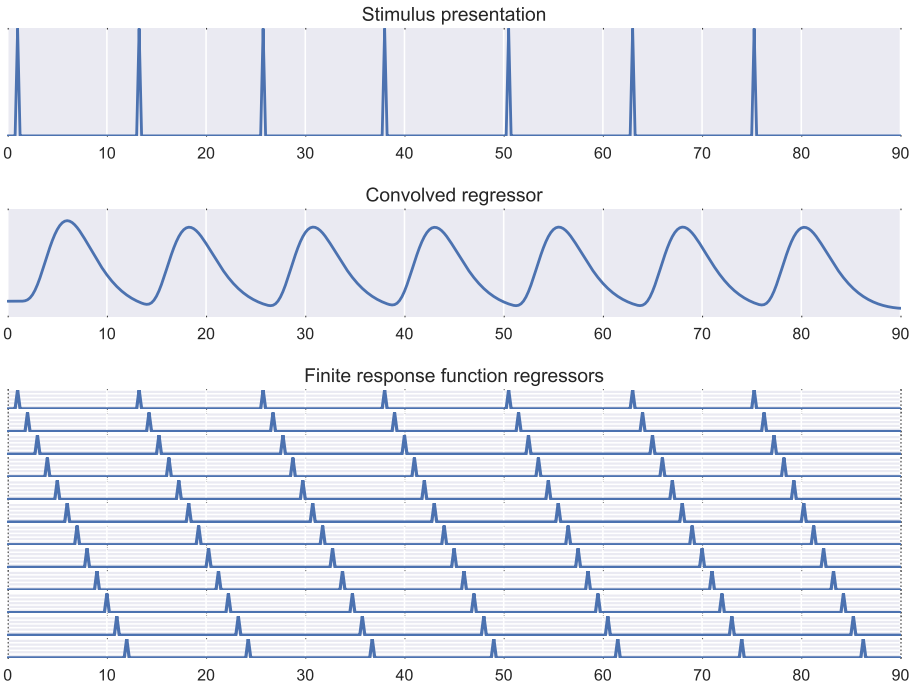


Figure 1: An experimental fMRI design with one stimulus type and two common models used in the GLM when modeling the resulting BOLD signal. The first panel shows the timeseries of the stimulus onsets. The second panel shows the stimulus onsets convolved with the double-gamma HRF, which can be interpreted as the expected BOLD signal if the measurement is related to the task. The parameter β in equation 1 with this model represents the amplitude of the signal related to the task.The third panel shows the FIR model, with each regressor a shifted version of the stimulus onsets. The β-parameters represent the amplitude of the HRF at specific time points following stimulus onset. Units on the x-axis are seconds. Units on the y-axis are removed, as these are meaningless and often rescaled to have unit height.

square matrix. With r_c the number of contrasts, there are two common ways to quantify the optimality of the design:

$$F = r_c/trace(C(X'WX)^{-1}C')\text{for A optimality}$$
$$F = det(C(X'WX)^{-1}C')^{-1/r_c}\text{for D optimality}$$

We denote F_e as the estimation efficiency if X is a FIR, and F_d as the detection power if X is a convolved design matrix.

2.2 Psychological measures of design optimality

Apart from the statistical concept of design efficiency, it is important to account for psychological factors that might render the experimental design invalid. The most important factor is predictability. For example in experiments addressing cognitive control, such as a stop-signal task, it is of the utmost importance that the trial type on any given trial cannot be easily predicted from the trial type on the previous trial, to avoid psychological confounding of the experiment. We quantify the optimality of the design in terms of confounding as:

$$F_c = \sum_{r=1}^{R}\sum_{i=1}^{Q}\sum_{j=1}^{Q} n_{ij}^r - (n-r)P_iP_j,$$

where n_{ij}^r is the number of trials of type i at timepoint t preceding a trial of type j at timepoint $t+r$. P_i is the proportion that trial should occur in the experiment. If $F_c = 0$, there are no unforeseen contingencies between trial types. The final optimality criterion controls the desired trial type frequencies: $F_f = \sum_{i=1}^{Q} |n_i - nP_i|$, with n_i the number of trials of type i.

2.3 Multi-objective criterion

To ensure comparability across different optimality criteria, we first rescale the the different optimality criterion to a scale of 0 to 1 as in Kao et al. (2009). To find the maximum F_d and F_e possible, we first run an optimisation with weights 1 for respectively F_d and F_e and weights 0 for the other optimality criteria. In the multi-objective criterion, the F_d and F_e scores are divided by their respective maximum to ensure scores between 0 and 1. For F_f and F_c, the score for the worst possible design (a design with only the least probable stimulus) is taken as the maximum score. Second, whereas larger F_e and F_d represent better design, the opposite is true for F_c and F_f. Therefore the scores for F_c and F_f are subtracted from 1. As such, the resulting optimality criteria can be written as

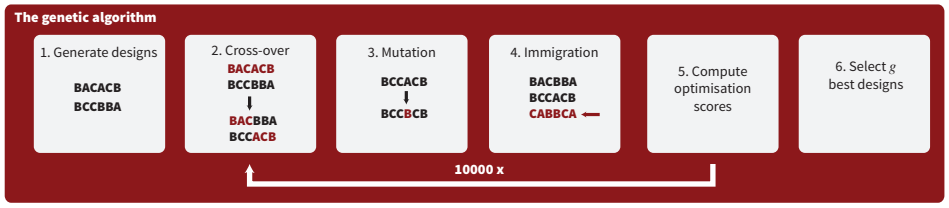


Figure 2: Graphical representation of the genetic algorithm. The examples in each step are pieces of experimental designs with 3 different trial types. In the example, the inter-trial interval is ignored.

$$
\begin{aligned}
F_i^* &= \frac{F_i}{\max(F_i)}, i = d, e; \\
&= 1 - \frac{F_i}{\max(F_i)}, i = c, f;
\end{aligned}
$$

As no design can ensure optimality in all four optimality criteria, the goal of any design optimisation depends on the researcher's goal of the experiment. Given prespecified weights w_i with $i = c, d, e, f, \sum_i w_i = 1, w_i \geq 0$, we define the weighted optimality criterion as: $F* = w_c F_c + w_d F_d + w_e F_e + w_f F_f$.

2.4 Genetic algorithm

A genetic algorithm is a method for solving optimisation problems inspired by natural selection in biological evolution. Contrary to classical optimisation algorithms, a genetic algorithm generates a population of points at each iteration. A graphic representation of the genetic algorithm with an fMRI example is shown in Figure 2. The steps of the genetic algorithm are.

1. Create G initial designs.
2. **Crossover.** Pair the best G/2 designs with each other.
3. **Mutation.** Randomly switch q% of all trials by random trial types.
4. **Immigration.** Add new random designs to the population.
5. **Natural selection.** Compute optimality scores and select G best designs
6. Repeat step 2-5 until a stopping rule is met.

3 Neurodesign, python module

3.1 Installation

NeuroDesign is available on **PyPi** and can be installed as:

```
pip install neurodesign
```

Next, we will give an introduction to the python module. For all functionality, please refer to the manual.

Figure 3: The basic layout of an experimental trial

3.2 Specifying the characteristics of the experiment

In a first step, the experiment should be described in the class called **experiment**. This contains general information, such as the number of stimuli and the duration of the experiment, but also more specific information, such as the model with which the inter trial intervals (ITI) are sampled. This function will generate the assumed covariance matrix, the drift function and the whitening matrix. All parameters are described in Table 1, while a graphical representation of components of an experiment are described in Figure 3. We define a simple experimental setup with 20 trials and 3 conditions, which we will use to exemplify the next functions:

```
from neurodesign import geneticalgorithm
EXP = geneticalgorithm.experiment(
   TR=1.2,
   n_trials=20,
   P = [0.3,0.3,0.4],
   C = [[1,-1,0],[0,1,-1]],
   n_stimuli = 3,
   rho = 0.3,
   stim_duration=1,
   ITImodel = 'uniform',
   ITImin = 2,
   ITImax=4
   )
```

3.3 Generating a design matrix

Within the defined experimental setup, we can now define a design matrix, develop the design matrix and compute the optimality scores using the class **design**. We use equal weights for the different optimality criteria for the weighted average optimality attribute. The only input required is the stimulus order, the ITI's and an object of class **geneticalgorithm.experiment**:

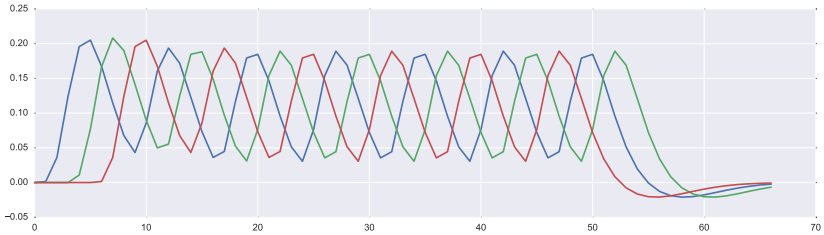
TR	The repetition time of the scanner.
n_stimuli	The number of different stimulus types or conditions.
P	The probabilities of each stimulus type.
C	The contrast matrix.
rho	The assumed autocorrelation coefficient
n_trials	The number of trials in the experiment. Either specify n_trials **or** duration
duration	The total duration (seconds) of the experiment. Either specify duration **or** n_trials
resolution (default = 0.1)	The resolution of the design matrix
t_pre (default = 0)	Duration (seconds) of the trial before the stimulus presentation (eg. fixation cross)
stim_duration	The duration (seconds) of the stimulus.
t_post (default = 0)	Duration (seconds) of the trial after the stimulus presentation.
maxrep (default = None)	The maximum number of times a stimulus is repeated consecutively.
hardprob (default = False)	True if the probabilities should be exactly the same as in P.
restnum (default = 0)	The number of trials between rest blocks
restdur (default = 0)	The duration (seconds) of a rest block
ITImodel	Which ITI model to sample from. Possibilities: 'fixed', 'uniform' or 'exponential'
ITImin	The minimum ITI (used with 'uniform' or 'exponential' ITImodel)
ITImean	The mean ITI (used with 'fixed' or 'exponential' ITImodel)
ITImax	The max ITI (used with 'uniform' or 'exponential' ITImodel)
confoundorder (default = 3)	The order to which confounding is controlled

Table 1: Arguments for object of class **geneticalgorithm.experiment**

```
from neurodesign import geneticalgorithm
DES1 = geneticalgorithm.design(
   order = [0,1,2,0,1,2,0,1,2,0,1,2,0,1,2,0,1,2,0,1],
   ITI = [2]*20,
   experiment=EXP
)
DES1.designmatrix(); DES1.FCalc(weights=[0.25,0.25,0.25,0.25])
```

Now using matplotlib, we can plot the convolved design matrix:

```
import matplotlib.pyplot as plt
plt.plot(DES1.Xconv)
```



We can now define a new design and compare both designs:

```
DES2 = geneticalgorithm.design(
    order = [0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1],
    ITI = [2]*20,
    experiment=EXP
)
DES2.designmatrix(); DES2.FCalc(weights=[0.25,0.25,0.25,0.25])
print("Ff of Design 1: "+str(DES1.Ff))
print("Ff of Design 2: "+str(DES2.Ff))
print("Fd of Design 1: "+str(DES1.Fd))
print("Fd of Design 2: "+str(DES2.Fd))
```

```
Ff of Design 1: 0.857142857143
Ff of Design 2: 0.428571428571
Fd of Design 1: 0.0870022296147
Fd of Design 2: 0.69476511364
```

As the second design ignores the presence of the third condition, the frequency optimality (F_f) is much worse. However, the blocked character of the design largely improves the detection power. The principles of the genetic algorithm, such as crossover, can be applied to the designs:

```
DES3,DES4 = DES1.crossover(DES2,seed=2000)
DES3.order
```

```
[0, 1, 2, 0, 1, 2, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
```

```
DES4.order
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1]
```

3.4 Optimal design using the genetic algorithm

To use the genetic algorithm to find the optimal design, we use the **geneticalgorithm.population** class, to denote the population of possible designs. All parameters are described in Table 2.

```
POP = geneticalgorithm.population(
    experiment=EXP,
    weights=[0,0.5,0.25,0.25],
    preruncycles = 10000,
    cycles = 10000,
    folder = "./",
    seed=100
    )
POP.naturalselection()
```

4 Neurodesign: the GUI

To make the methods more publicly available, we have created a graphic user interface running in a web-application. The back-end of the application is written in python and uses the python module neurodesign described above, the front-end is generated using django, and the application is deployed through a multi-container docker environment on Amazon Web Services.

There are 5 crucial windows of the GUI: main input, contrasts and probabilities, review, console and settings. A part of the main input window is shown in Figure 4, which has fields for most parameters from Table 1. Only the parameters P and C are asked in the second window ('Contrasts and probabilities'). The review window shows all parameters and also prints out the default settings for the genetic algorithm. These parameters, presented in Table 2, can be adjusted in the settings window. The console allows for the optimizations

experiment	The experimental setup of the fMRI experiment (of class neurodesign.experiment)
G (default = 20)	The size of each generation
R (default = [0.4,0.4,0.2])	The rate with which the orders are generated from (a) blocked designs, (b) random designs and (c) m-sequences
q (default = 0.01)	The percentage of mutations in each generation
weights	The weights attached to [F_e, F_d, F_f, F_c]
I (default = 4)	The number of immigrants in each generation
preruncycles	The number of pre-run cycles to find the maximum value of F_e and F_d
cycles	The number of cycles in the optimisation
seed	The random seed for the optimisation
Aoptimality (default = True)	Optimises A-optimality if true, else D-optimality
convergence (default = 1000)	After how many stable iterations is there convergence
folder	The local folder to save the output
outdes (default = 3)	The number of designs to be saved

Table 2: Arguments for object of class **geneticalgorithm.population**

to be started, stopped and followed. When a design optimization is started, the user receives an email with a link to the console where the optimization can be followed (Figure 5). Once the optimization is finished, a zip file can be downloaded containing a chosen number of designs. Each design contains the onsets for each stimulus, a report with design diagnostics (such as collinearity among regressors, see Figure 3), and a script. The script can be used for future reference, or for regenerating the designs locally.

5 Discussion

5.1 Default settings

Both the initialisation of the experiment, represented by the class experiment in the python module and the main input window in the GUI, and the genetic algorithm, represented by the class population in the python module and the settings window in the GUI, have some default settings. The default settings of the python module, shown in Tables 1 and 2, are optimised for a good optimisation, while the GUI's default settings are optimised for short optimisation duration. While the default settings for the GUI can lead to a sub-optimal design, the user is warned (with a big red textblock at the top of the page) that the settings should be changed if the results will be used for a good optimisation. We have chosen these sub-optimal defaults for the GUI to provide a fast run through for first time users, as well as to avoid memory and CPU overload on the server end. For the experiment, we assume a priori that there are no rest blocks and that the trial only consists of stimulation (no fixation cross etc.). There is by default no limit on the maximum number of times a stimulus can be

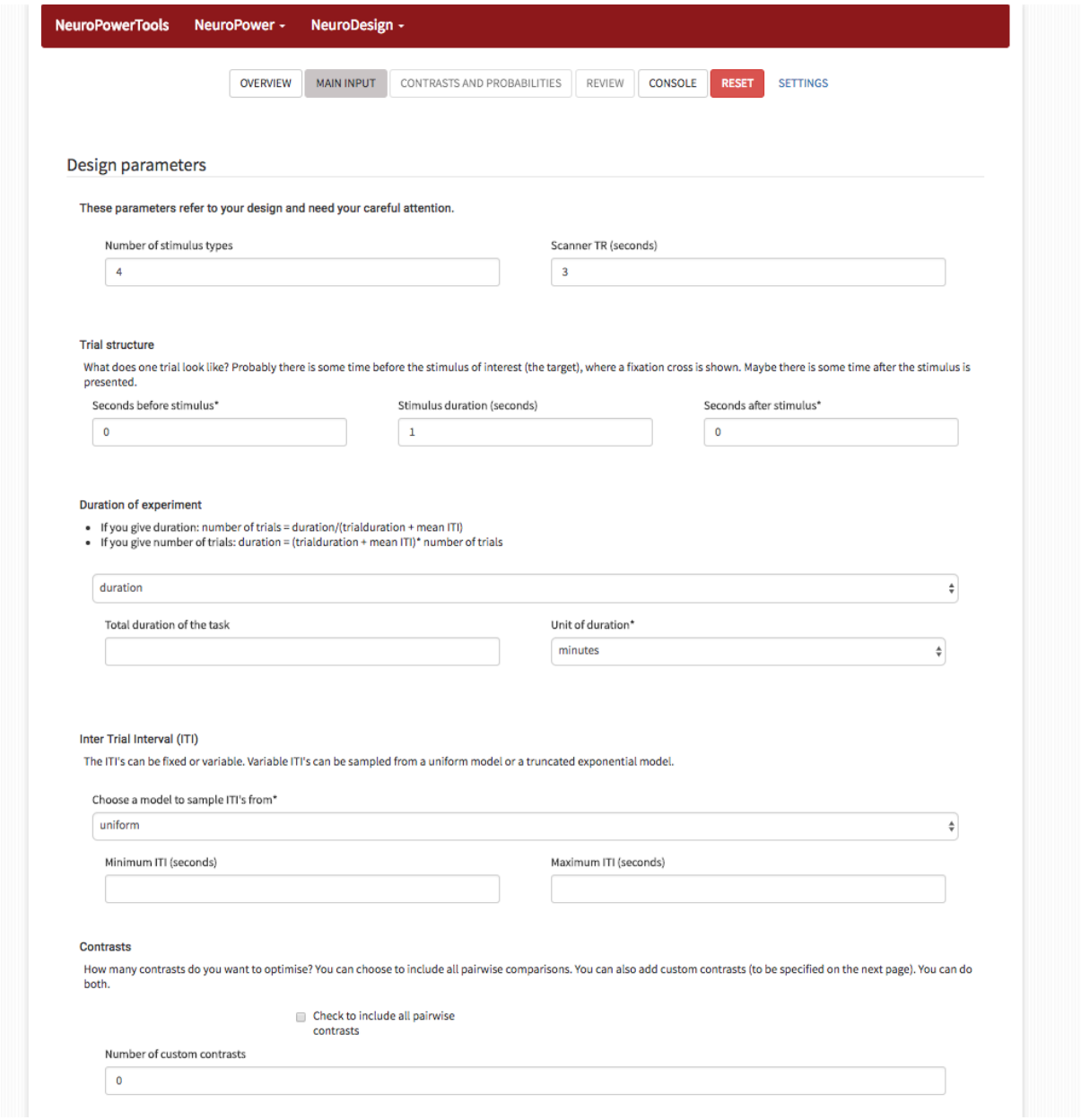


Figure 4: Part of the input window for the GUI.

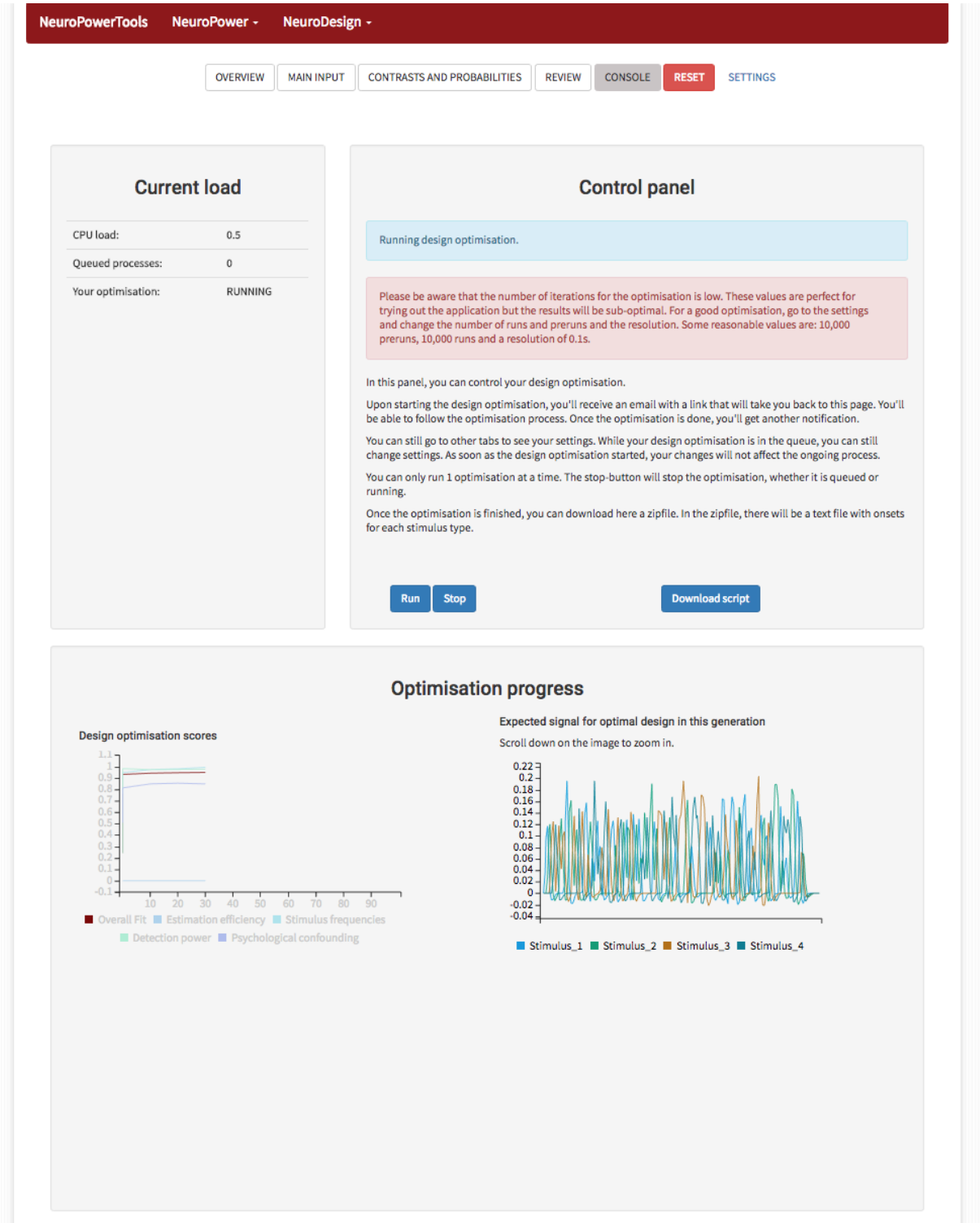


Figure 5: Screenshot of the console where the optimisation can be followed. Every 10 generations, the design is updated with the latest score and the best design.

Selected designs

The following figure shows in the upper panel the optimisation score over the different generations. Below are the expected signals of the three best designs from different families. Next to each design is the covariance matrix between the regressors, and the diagonalmatrix with the eigenvalues of the design matrix.

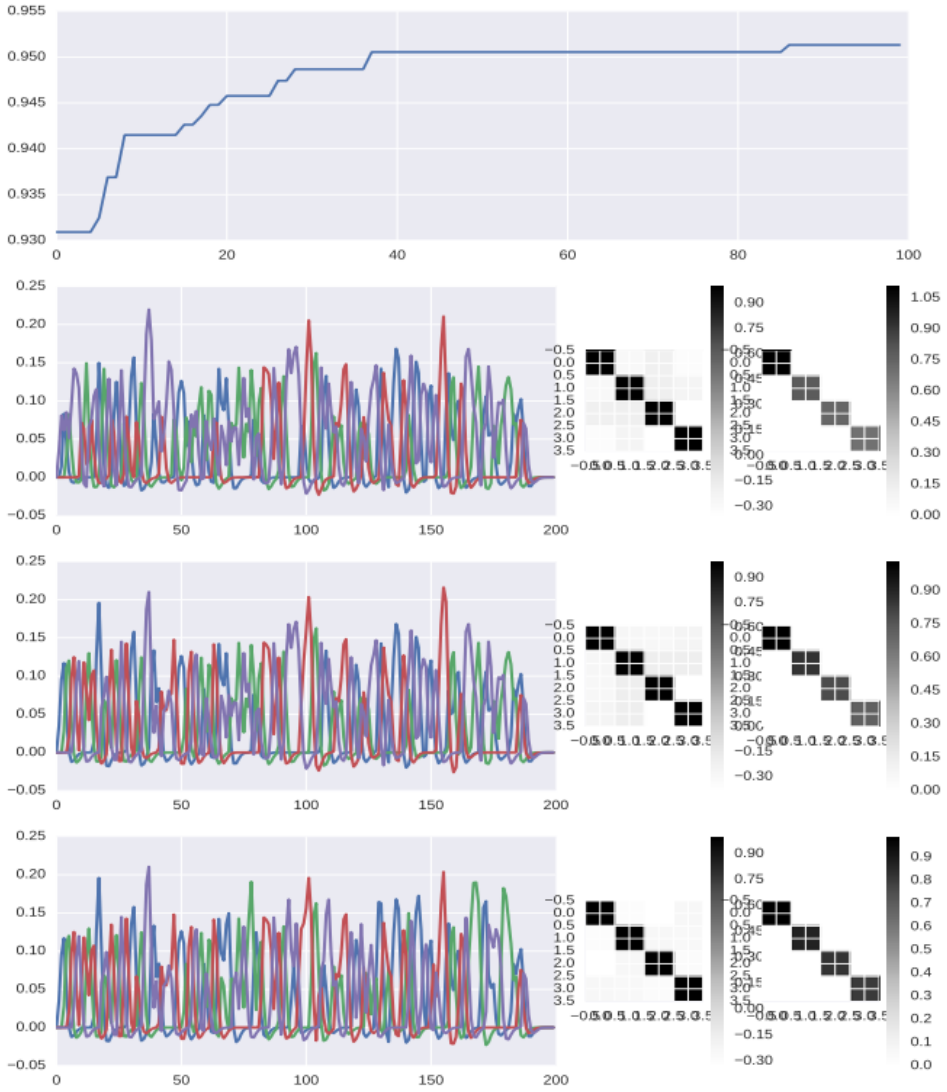


Figure 6: Screenshot of the report describing the optimisation and the best 3 designs from the optimisation.

repeated, and the stimulus frequency is not controlled with a hard limit. The default resolution in the python module is 0.1 seconds, while in the GUI it is 0.25 seconds. For the genetic algorithm, the default settings are as follows. The optimisation calculates the A-optimality. In each generation, the percentage of mutations is 1%, the number of immigrants is 4 designs and the size of each generation is 20 designs. When generating new designs, there are 40% blocked designs, 40% random designs and 20% m-sequences. Convergence is reached when the score is stable for 1000 generations. There are no default settings on the number of cycles in the python module, while the GUI runs by default 10 cycles to find the maximum F_d and F_e and 100 cycles for the optimisation (again with a clear message that this should be increased for optimal results).

5.2 Reproducibility

In line with the recent effort to make neuroimaging research fully reproducible, this application makes it possible to track the exact source of each design. Low level reproducibility is provided by making a script available for download with which the optimisation can be regenerated. Running this script in python, given that the required libraries are installed, will repeat the analysis. However, this script will repeat the analysis but does not guarantee the same results as the specific configuration of the computer on which the analysis is run can influence the results.

Higher level reproducibility, that guarantees replicability not only of the analysis but also of the results, is possible with the use of docker containers, which is a small piece of software that emulates a given computational configuration (operating system, libraries, python packages,...). Based on the model presented by BIDS-apps (Gorgolewski et al., 2016), our analyses run in docker containers that are open-source and available for download at https://hub.docker.com/r/neuropower/neuropower/. Running the following command in a terminal will replicate the analysis that has been performed through the GUI.

```
docker run -v /location_where_the_script_is/:/local \\
-it neuropower/neuropower python /local/name_of_the_script.py
```

This use of docker containers is not only well suited for reproducibility of the GUI, but also allows the replication of results from a python script (given that a random seed is set).

6 Conclusion

We present a toolbox for optimizing fMRI designs. The toolbox is an extension of currently available toolboxes, allowing for more complex design and better control and optimization of timing of stimuli. The toolbox is available through

different modalities: a user-friendly GUI accessible at www.neuropowertools.org and a python package. The code is available on www.github.com/users/neuropower.

7 Acknowledgements

This work was supported by the Laura and John Arnold Foundation. J.D. has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 706561.

References

Buracas, G. T. and G. M. Boynton
2002. Efficient design of event-related fMRI experiments using m-sequences. *Neuroimage*, 16(3 Pt 1):801–813.

Button, K. S., J. P. A. Ioannidis, C. Mokrysz, B. A. Nosek, J. Flint, E. S. J. Robinson, and M. R. Munafò
2013. Power failure: why small sample size undermines the reliability of neuroscience. *Nat. Rev. Neurosci.*, 14(5):365–376.

Dale, A. M.
1999. Optimal experimental design for event-related fMRI. *Hum. Brain Mapp.*, 8(2-3):109–114.

Durnez, J., J. Degryse, B. Moerkerke, R. Seurinck, V. Sochat, R. Poldrack, and T. Nichols
2016. Power and sample size calculations for fMRI studies based on the prevalence of active peaks.

Durnez, J., B. Moerkerke, and T. E. Nichols
2014. Post-hoc power estimation for topological inference in fMRI. *Neuroimage*, 84:45–64.

Eklund, A., T. E. Nichols, and H. Knutsson
2016. Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. *Proc. Natl. Acad. Sci. U. S. A.*, 113(28):7900–7905.

Gorgolewski, K. J., F. Alfaro-Almagro, T. Auer, P. Bellec, M. Capota, M. Mallar Chakravarty, N. W. Churchill, R. Cameron Craddock, G. A. Devenyi, A. Eklund, O. Esteban, G. Flandin, S. S. Ghosh, J. Swaroop Guntupalli, M. Jenkinson, A. Keshavan, G. Kiar, P. R. Raamana, D. Raffelt, C. J. Steele, P.-O. Quirion, R. E. Smith, S. C. Strother, G. Varoquaux, T. Yarkoni, Y. Wang, and R. A. Poldrack
2016. BIDS apps: Improving ease of use, accessibility and reproducibility of neuroimaging data analysis methods.

Hayasaka, S., A. M. Peiffer, C. E. Hugenschmidt, and P. J. Laurienti
2007. Power and sample size calculation for neuroimaging studies by non-central random field theory. *Neuroimage*, 37(3):721–730.

Ioannidis, J. P. A.
2005. Why most published research findings are false. *PLoS Med.*, 2(8):e124.

Kao, M.-H., A. Mandal, N. Lazar, and J. Stufken
2009. Multi-objective optimal experimental designs for event-related fMRI studies. *Neuroimage*, 44(3):849–856.

Mumford, J. A. and T. E. Nichols
2008. Power calculation for group fMRI studies accounting for arbitrary design and temporal autocorrelation. *Neuroimage*, 39(1):261–268.

Open Science Collaboration
2015. PSYCHOLOGY. estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716.

Wager, T. D. and T. E. Nichols
2003. Optimization of experimental design in fMRI: a general framework using a genetic algorithm. *Neuroimage*, 18(2):293–309.