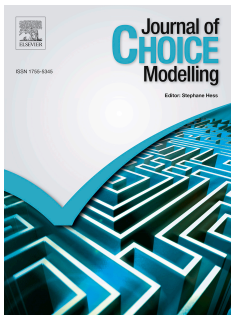
Accepted Manuscript



New software tools for creating stated choice experimental designs efficient for regret minimisation and utility maximisation decision rules

Sander van Cranenburgh, Andrew T. Collins

PII: S1755-5345(18)30094-0

DOI: https://doi.org/10.1016/j.jocm.2019.04.002

Reference: JOCM 166

To appear in: *Journal of Choice Modelling*

Received Date: 8 August 2018

Revised Date: 2 April 2019

Accepted Date: 2 April 2019

Please cite this article as: van Cranenburgh, S., Collins, A.T., New software tools for creating stated choice experimental designs efficient for regret minimisation and utility maximisation decision rules, *Journal of Choice Modelling* (2019), doi: https://doi.org/10.1016/j.jocm.2019.04.002.

New software tools for creating stated choice experimental designs efficient for regret minimisation and utility maximisation decision rules

Sander van Cranenburgh[1]
Andrew T. Collins[2]

[1]Delft University of Technology, Faculty of Technology, Policy & Management, Transport and Logistics Group
[2]The University of Sydney, Institute of Transport and Logistics Studies

Abstract
At the time of creating an experimental design for a stated choice experiment, the analyst often does not precisely know which model, or decision rule, he or she will estimate once the data are collected. This paper presents two new software tools for creating stated choice experimental designs that are simultaneously efficient for regret minimisation and utility maximisation decision rules. The first software tool is a lean, easy-to-use and free-of-charge experimental design tool, which is dedicated to creating designs that incorporate regret minimisation and utility maximisation decision rules. The second tool constitutes a newly developed extension of Ngene – a widely used and richly featured software tool for the generation of experimental designs. To facilitate the use of the new software tools, this paper presents clear worked examples. It focusses on practical issues encountered when generating such decision rule robust designs, such as how to obtain priors and how to deal with alternative specific parameters. Furthermore, we analyse the robustness of the designs that we created using the new software tools. Our results provide evidence that designs optimised for one decision rule can be inefficient for another – highlighting the added value of decision rule robust designs.

Keywords: Efficient design; Stated choice; Random Regret Minimisation; Decision rules; Software.

1 Introduction

Stated Choice (SC) experiments are widely used to acquire understanding of choice behaviour in a variety of research fields, including but not limited to transportation, marketing, and health and environmental economics (Louviere et al. 2000; Street and Burgess 2007; Rose and Bliemer 2009; de Bekker-Grob et al. 2012). In SC experiments respondents are presented with choice tasks involving two or more hypothetical alternatives, which are described by a set of attributes and attribute levels. Respondents are asked to assess the alternatives and make a choice, typically of their most preferred alternative. Prior to the SC experiment the analyst creates the experimental design, which involves allocating attribute levels to the alternatives of each choice task. There are different approaches to generate experimental designs, of which so-called efficient designs are most common at present. Efficient designs aim to maximise the information obtained from the SC data, resulting in more reliable parameter estimates for a given number of observations (Rose and Bliemer 2009; Kessels et al. 2011).

To date, research and software for experimental design have almost exclusively been based on the (often implicit) assumption that decision-makers make choices using a (linear-additive) Random Utility Maximisation (RUM) decision rule. However, a growing number of studies have found overwhelming evidence that decision-makers may opt for other types of decision rules when making choices (Kivetz et al. 2004; Hess et al. 2012; Leong and Hensher 2012; Guevara and Fukushi 2016; Hancock et al. 2018; Van Cranenburgh and Alwosheel 2019). In light of this, very recently a method to create efficient experimental designs for one alternative decision rule, namely Random Regret Minimisation (RRM), has been proposed (Van Cranenburgh et al. 2018). RRM models postulate that decision-makers choose the alternative that provides them with minimum regret, which is caused by the need to trade-off attributes of alternatives during the decision making process (Chorus 2010). Unlike its RUM counterpart, RRM models feature a particular reference-dependent type of semi-compensatory behaviour.

One particularly important result of the study by Van Cranenburgh et al. (2018) is that they find that designs that are efficient for estimating RUM models can be highly inefficient for estimating RRM models, and vice versa. Therefore, they advocate taking multiple decision rules into account when creating efficient experimental designs. To create such 'decision rule robust designs' they propose to use a model averaging approach, akin to the approach taken by Rose et al. (2009) to account for uncertainty regarding the model specification (e.g., multinomial logit, mixed logit). But, although the theory to devise designs which are robust toward the uncertainty on the side of the analyst regarding the underlying decision rule has recently been established, the burden to actually generate these designs is currently high: it requires extensive software coding on the side of the analyst.

This paper aims to lower the burden for analysts who wish to create SC experimental designs that are simultaneously efficient for estimating RUM and RRM models. In particular, it presents two software tools in which such decision rule robust designs can be generated. The first software tool is called *Robust Design Generator* (RDG). RDG is a lean, easy-to-use and free-of-charge experimental design tool, running in a MATLAB environment. RDG is confined to the design of unlabelled experiments with three alternatives. The second tool constitutes a newly developed extension of Ngene. Ngene is an established, highly versatile commercial software dedicated to the design of SC experiments (ChoiceMetrics 2018). To facilitate creating decision rule robust efficient designs using the two new software tools, this paper presents clear and worked out examples. It focusses on practical issues encountered when generating such efficient designs, such as how to obtain priors and how to deal with alternative specific parameters.

The remaining part of this paper is organised as follows. Section 2 briefly revisits efficient design theory for RRM models. Section 3 shows how to create efficient designs using the two software tools. In this section the robustness of the created designs is also analysed. Finally, section 4 provides a conclusion and discusses new avenues for further research.

2 RRM efficient design theory

This section briefly revisits the theory behind efficient designs for estimating RRM models. Theory on efficient designs for estimating RUM models is widely available in the literature (see e.g. Rose and Bliemer 2009). Although generating efficient designs is in many respects very similar for both RRM and RUM models, this section details a number of key differences between the two modelling paradigms that need to be discussed. This includes how to obtain adequate priors for RRM designs (Section 2.2), and how the analyst can deal with non-generic variables (e.g. labels) in RRM designs (Section 2.3). Additionally, the creation of designs that are robust towards uncertainty on the side of the analyst regarding the decision rule are discussed in Section 2.4.

2.1 Efficient designs for estimating RRM models

RRM models are descriptive models of choice. A particular feature of RRM models – as compared to the normative linear-additive RUM model – is that they are able to capture some frequently observed behavioural phenomena. One notable behavioural phenomenon that can be captured by RRM models is the compromise effect. The compromise effect is the behavioural notion that – from a utilitarian perspective – the compromise alternative[1] receives a higher market share than one would expect based on its attribute levels (Chorus and Bierlaire 2013). The idea is that by accounting for this behavioural phenomenon, RRM models may be able to better explain and predict choice behaviour.

RRM models postulate that decision-makers choose the minimum regret alternative (Chorus 2010). Regret is experienced by the decision-maker when a competitor alternative *j* outperforms the considered alternative *i* with regard to one or more attributes *m*. The total regret associated with an alternative is – in most RRM model specifications – the sum of all the pairwise regrets that are associated with bilaterally comparing the considered alternative with the other alternatives in the choice set. The most widely used mathematical form of RRM models is shown in Equation 1, where RR_{in} denotes the random regret for decision-maker *n* who considers alternative *i*, R_{in} denotes the observed, or systematic, part of regret, and ε_{in} denotes the error term which captures the unobserved part of regret. r_{ijmn} is the so-called attribute level regret function. This function maps the difference between the levels of attributes *m* of the competitor alternatives *j* and the considered alternative *i* onto regret. Different specifications of this function lead to different types of RRM models (see Van Cranenburgh and Prato (2016) for an overview of RRM models, and their specifications). All specifications have in common that they have convex attribute level regret functions. As a result of that, RRM models predict that having a (very) poor performance on one attribute (which causes much regret) cannot necessarily be compensated for by having a (very) strong performance on another attribute. A compromise alternative will only generate modest levels of regret since it has no particularly poor performance on any of the attributes. As a result, RRM models predict that the compromise alternative is relatively attractive

[1] Compromise alternatives have an intermediate performance on each or most attributes rather than having a poor performance on some attributes and a strong performance on others.

in terms of minimising regret (see Chorus 2012 for a more extensive explanation on how RRM models capture the compromise effect).

$$RR_{in} = R_{in} + \varepsilon_{in} \text{ where } R_{in} = \sum_{j \neq i} \sum_{m} r_{ijmn} \text{ where } r_{ijmn} = f\left(\beta_m, x_{jmn} - x_{imn}\right) \quad \text{Equation 1}$$

In the remainder of our paper, we use the P-RRM model (Van Cranenburgh et al. 2015a). The P-RRM model is a special case of the generic and flexible µRRM model[2], and has a cornerstone interpretation within the RRM modelling paradigm as it postulates no 'rejoice' (the opposite of regret). In other words, it yields the strongest regret minimisation behaviour, i.e., the highest level of regret aversion which is possible within the RRM modelling framework. Its attribute level regret function is given by $r_{ijmn} = \max\left(0, \beta_m \left[x_{jmn} - x_{imn}\right]\right)$. This piece-wise linear function consists of a max operator, which at first sight may seem undesirable from a numerical optimisation perspective. However, when the signs of the taste parameters are known to the researcher – as is most often the case – the P-RRM model becomes linear-additive (Equation 2). This, in turn, makes this model particularly attractive for creating efficient designs.

$$R_{in}^{P-RRM} = \sum_{m} \beta_m x_{imn}^{P-RRM} \qquad where\ x_{imn}^{P-RRM} = \begin{cases} \sum_{j \neq i} \max\left(0, x_{jmn} - x_{imn}\right) & if\ \beta_m > 0 \\ \sum_{j \neq i} \min\left(0, x_{jmn} - x_{imn}\right) & if\ \beta_m < 0 \end{cases} \quad \text{Equation 2}$$

Van Cranenburgh et al. (2018) derive the Fisher information matrix for this RRM model (in MNL form), which is essential to the creation of efficient designs, see Equation 3. Using the Fisher information matrix the Asymptotic Variance Covariance (AVC) matrix, denoted Ω, can be derived (Equation 4). As can be seen, for the P-RRM MNL model the Fisher information matrix *I* is not a function of the actual choices *Y*. As such, the AVC matrix can analytically be computed in a straightforward way, just like for linear-additive RUM MNL models (Huber and Zwerina 1996) and linear-additive RUM Nested Logit models (Bliemer et al. 2009). From a computational point of view, this is a highly desirable feature, given that to find an efficient design typically many thousands of candidate designs have to be evaluated.

$$I(\beta \mid X) = -E_Y\left(\frac{\partial^2 \log L(\beta \mid X, Y)}{\partial \beta \partial \beta}\right) = \sum_{s=1}^{S} \sum_{j=1}^{J} x_{jm_1 s}^{P-RRM} P_{js} \left(x_{jm_2 s}^{P-RRM} - \sum_{i=1}^{J} x_{im_2 s}^{P-RRM} P_{is}\right) \quad \text{Equation 3}$$

$$\Omega(\beta \mid X) = \left(I(\beta \mid X)\right)^{-1} \quad \text{Equation 4}$$

[2] The µRRM model is a generalisation of the RRM model proposed by Chorus (2010). In this model, an additional parameter µ is estimated, jointly with the taste parameters. This parameter captures the extent to which losses loom larger than equivalently sized gains.

To compute the efficiency of a design, different measures of statistical efficiency can be used (Kessels et al. 2006). In the context of this paper we mainly focus on one measure of statistical efficiency: the D-error statistic (D-efficiency). The D-error statistic is the most commonly used measure of efficiency in experimental design practice, and is calculated by taking the determinant of the AVC matrix, see Equation 5.

$$D\text{-error} = \det\left[\Omega\left(X, \beta\right)\right] \qquad \text{Equation 5}$$

2.2 Priors for P-RRM models

Prior parameters are an essential ingredient for creating efficient designs. Prior parameters essentially inform the design algorithm on where the trade-off points of decision-makers can be expected to be. By doing so, they allow the creation of designs which provide a maximum amount of information on the sizes and signs of the estimable parameters. However, regardless of whether one generates efficient designs for RUM or RRM, it is important to set the priors carefully. There is extensive literature – in the context of RUM efficient designs – showing that when accurate priors are used efficient designs can result in substantial statistical benefits, as compared to orthogonal or random designs (e.g. Bliemer and Rose 2011). But, ill-chosen priors may undermine the sole purpose of creating efficient designs as they may lead to statistical inefficient designs (Ferrini and Scarpa 2007; Walker et al. 2018). The most common way to obtain prior parameters is by conducting a pilot study. For RUM and RRM efficient designs alike, a pilot study may provide valuable insights on what are good priors. However, it should be noted that RRM parameters are less transferable from one design set-up to another. Unlike RUM models, RRM models are context dependent models of choice. This means that they are sensitive towards the composition of the choice task. By extension, RRM model parameters are context dependent. The implication is that the analyst has limited freedom to adjust the design dimensions of the experiment, such as changing the ranges of the attribute levels or changing the number of alternatives in a choice task, after having conducted the pilot study, without jeopardising the reliability of the obtained priors from the pilot study.

In the absence of a pilot study, a second best way to obtain RRM priors is based on the literature. However, the vast majority of the discrete choice modelling literature involves RUM models, not RRM model. Although no theoretical relation exist between RUM and RRM estimates, RUM modelling results can be used to obtain proxies for RRM priors. From the numerous studies comparing RUM and RRM models, we know that when RUM and RRM models are estimated on the same data (1) the ratios of the parameters are fairly stable across both models, and (2) RRM parameters are roughly $2/J$ times smaller than their RUM counterparts, where J denotes the number of alternatives in choice set (Van Cranenburgh et al. 2015b). Accordingly, when consulting the RUM literature to set priors for RRM efficient designs we recommend to rescale the RUM parameters using a factor of $2/J$. Note that this factor also has been used to estimate

RRM models on data sets in which the number of alternatives that are available to decision-makers varied across choice observations (Van Cranenburgh and Chorus 2018).

Regardless of how the prior parameters are obtained, we believe it is generally good practice to account for uncertainty regarding the prior parameters using so-called Bayesian efficient designs (Sandor and Wedel 2001). These designs explicitly take into account the uncertainty regarding the prior parameters. In Bayesian designs the analyst specifies the prior as a random number rather than as a fixed number. The efficiency of a Bayesian design is evaluated as the mean[3] efficiency across all draws for the randomly drawn prior parameters. Hence, for Bayesian efficient designs the analyst still needs to define the priors (including the shapes of their distributions). But, the idea is that these design choices are less sensitive towards misspecification. In the literature, a number of papers provide guidance on how to obtain Bayesian priors, see e.g. Kessels et al. (2008); and Bliemer and Collins (2016).

Finally, zero priors are not permissible for RRM efficient designs (unlike for RUM efficient designs). Due to the min and max operators in the attribute level regret function of the P-RRM model (Equation 2), the second order derivative – which is used to create efficient designs – does not exist at $\beta = 0$. This is a trivial problem in practice. In case an analyst aims to generate a non-Bayesian efficient design and has no particular expectation regarding the size of a parameter but some clue about the sign, we simply recommend using a very small value (positive or negative, depending on the sign expectation). In case an analyst aims to generate a Bayesian efficient design[4], they may simple use a random parameter, with a mean of zero (e.g. a uniform distribution between -1 and 1).

2.3 Alternative specific variables

It only makes sense to estimate RRM models when attributes are generic, i.e. attributes are shared across alternatives. After all, behavioural phenomena which the RRM model aims to capture – like the compromise effect – may arise only for generic attributes. However, in some instances, alternatives in SC experiments consist of attributes that are specific to particular alternatives. For instance, in a mode choice between car, bus and train in the transportation field, having or not having access to WIFI may only be applicable to the train alternative. RUM and RRM treatment of such variables – which are binary in the difference across alternatives – is mathematically equivalent, see Appendix B for a formal proof. Note that for RRM models which have a logarithm in the attribute level regret function (i.e. $r_{ijm} = \ln\left(1+\exp\left(\beta_m\left[x_{jm} - x_{im}\right]\right)\right)$) RUM and RRM treatment of variables which are binary in the difference across alternatives is also mathematically equivalent, apart from a rescaling which has no impact on the statistical efficiency or model fit (Chorus 2012; Hess et al. 2014). Therefore, when creating efficient designs for RRM models which involve a combination of generic and alternative specific attributes no special attention needs to be paid to alternative specific variables.

[3] Note that sometimes not the mean, but the median, minimum or maximum value is used in Bayesian design.
[4] Only applicable to Ngene.

2.4 Efficient designs robust for decision rule uncertainty

Overwhelming empirical evidence shows that humans use a wide variety of decision rules when making choices. As a result, the analyst usually does not know prior to conducting the choice experiment what will be the prevailing decision rule. This is a motivation to take multiple decision rules into account when creating efficient designs, instead of just one. This is especially so as it is known that efficient designs which are optimised having one particular model (with a set of priors) in mind can be highly inefficient when another model is being estimated once the data are collected. Rose et al. (2009) have shown this in the context of different RUM model specifications, and more recently Van Cranenburgh et al. 2018 have shown this in the context of RUM and RRM decision rules. Both papers therefore advocate the use of designs which are robust towards multiple models (be it model specifications, or embedded decision rules), by means of minimising a composite D-error which is based on a weighted sum of D-errors associated with different models, see Equation 6. In the context of decision rules, a composite efficiency measure takes into account the probability of each decision rule being the model that the analyst will estimate, once the data are collected. In case the analyst has no particular expectation regarding the most likely decision rule in the data (and thus the models he or she will ultimately estimate and report), equal weights w_r may be given to the considered decision rules (denotes r). Equal weights mean that the D-error of each decision rule roughly contributes equally to the composite efficiency measure. The actual contribution to the composite D-error may not exactly equal w_r as the D-error of one model may be larger (or smaller) than the D-error of the other model. In case the analyst has expectations regarding the most likely decision rule in the data, the analyst may adjust the weights to reflect these expectations. Like with prior parameters, such expectations may be based on results from pilot surveys, or obtained from the literature.

$$D_{composite} = \sum_{r=1...R} w_r \cdot D_r\text{-error} \quad where \quad \sum_{r=1...R} w_r = 1 \qquad \text{Equation 6}$$

3 New software tools

Two software tools have been developed to create efficient designs that are robust for estimating RUM and RRM models. The first tool, Robust Design Generator, is a lean, fast and easy-to-use experimental design tool, working in a MATLAB environment. RDG is free-of-charge, but provides relatively limited flexibility. In particular, it is confined to three alternative choice sets only. The second tool is an implementation in the widely used Ngene software package. Ngene software has many features for the generation of experimental designs and is far more flexible than RDG. But, Ngene is commercial software, and hence needs to be purchased. By providing two platforms (albeit not fully equivalent), we aim to serve a broad group of users. Below, we discuss – for each tool – how to create designs on the basis of a number of worked examples.

3.1 Robust Design Generator

RDG is dedicated to creating designs aimed for investigating RRM and RUM decision rules. More specifically, RDG allows the analyst to create experimental designs that are efficient for P-

RRM MNL and RUM MNL models, as well as for a mixture thereof. RDG software can be downloaded from www.advancedRRMmodels.com.

There are two ways to use RDG:

1. **As a MATLAB app**, which runs within the MATLAB environment. Advantages of running RDG as MATLAB app are that it allows the analyst to extend and build upon the software code and that it is very easy to install. This approach requires the analyst to have installed MATLAB R2017 (or higher).
2. **As a stand-alone application**. This approach does not require a MATLAB license. Therefore, in this form the tool can be used by anybody, without the need to purchase any commercial software. Installation requires about 2GB hard drive space. See www.advancedRRMmodels.com for further installation details.

Note that the MATLAB app and the stand-alone application are essentially the same: the only difference being the need for a MATLAB license and the ability to modify the underlying code.

3.1.1 Layout and use

The layout of the RDG tool is shown in Figure 1. The RDG tool has a single pane layout, with two panels. The left-hand side panel is where the analyst defines the design set-up (i.e. the design dimensions). It consists of input fields for the attribute levels and the prior parameters; it contains push buttons to select the model to optimise for, as well as a push button to start and reset the search algorithm. Also the left-hand side panel is used for monitoring the optimisation. At the bottom of the panel, progress messages are provided. The D-error of the last run, the lowest D-error found so far, and the S-estimate associated with the best design found are reported. The S-estimate is a theoretical lower bound for the sample size required for finding a statistically significant parameter estimate for a parameter, conditional on the prior for that parameter being accurate (Rose and Bliemer 2013). Thereby, the measure provides a useful indication to the analyst on the minimum required sample size, prior to the actual data collection. The right-hand side panel is where the main output and the design are displayed, after invoking the software to stop searching using the 'STOP' button. Also, general statistics on the design are reported in the output text area.

To construct an efficient design, the analyst performs 5 steps. Firstly, the analyst enters the number of choice tasks of the design. Secondly, the analyst inserts the attribute levels for each attribute. The interface allows for a maximum of 6 levels per attribute, and up to four attributes. Thirdly, the analyst specifies the values of the prior parameters. To account for uncertainty regarding the prior parameters, the analyst can tick the boxes to search for a Bayesian D-efficient design. When a box is ticked for a prior parameter, the software will create 100 i.i.d. draws from a symmetric triangular distribution. The created triangular distribution has a domain between zero and two times the value of the prior entered (hence, the mean of the triangular distribution is equal the prior). Of course, other types of distributions could have been chosen. We decided to implement this distribution because of its simplicity: no additional scale or shape parameters are needed and all its mass is either in the positive, or negative domain – which is a desirable feature given the discontinuity of the P-RRM model at $\beta = 0$.

Note that creating Bayesian D-efficient designs is considerably slower than creating non-Bayesian efficient designs. Therefore, it is good practice to create experimental designs using non-Bayesian priors initially, to ensure that the design priors are appropriately scaled. Fourthly, the analyst selects the model to optimise for, by selecting a model button. The analyst has three options: RUM-MNL, P-RRM-MNL, or a mixture of both (where equal weight is given to both models). Fifthly, when the analyst has entered everything correctly, the analyst may start the search algorithm by pressing the 'START' button. The status lamp will turn orange, indicating the algorithm is searching the solution space.

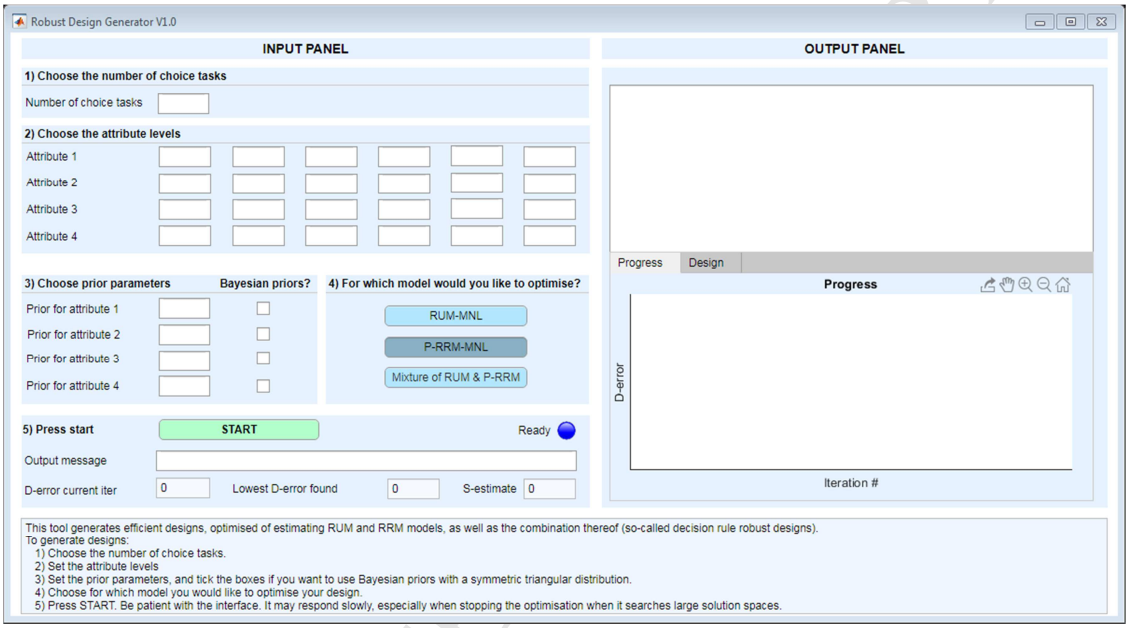


Figure 1: RDG user interface

The algorithm will first create the candidate set of choice tasks, which may take a few seconds to tens of minutes, depending on the dimensions of the design set-up. Choice tasks with dominant alternatives are automatically removed from the candidate set. After that, it will start searching the solution space using a Modified Fedorov algorithm (Fedorov 1972), and will indicate this in the output message box. The Fedorov algorithm starts by taking a randomly drawn design (this is termed 'an iteration'), and continues by replacing choice tasks in the design with those from the candidate set and evaluating its impact (see e.g. Van Cranenburgh and Bliemer 2018). After having exhausted the first series of replacements, a first efficient design is found. This will prompt the first D-error at the bottom of the input panel, under 'D-error current iter'. Also, a 'STOP' button becomes visible, next to the 'START' button. When this button is selected by the analyst, the search algorithm stops and the best design that has been found so far will be shown in the output panel. It is good practice to let the search algorithm do at least a few iterations. The analyst can monitor the number of iterations and the lowest D-error that has been found, at the bottom of the input panel as well as in the lower right the progress plot.

Note that the interface may be slow to respond, when the optimisation is stopped when it searches large solution spaces (i.e. when the dimensionality of the design is large). The reason has to do with the fact that the algorithm first creates the full candidate set, consisting of all non-dominating choice tasks and then starts replacing choice tasks from the design with the candidate choice tasks. The algorithm can only be stopped after having finished a full series of replacements (i.e. an iteration). The time required to perform one such iteration increases with the dimensionality of the design, which is why the interface can be slow to respond. However, a key advantage of this implementation (i.e. working with the full candidate set) is that from iteration 1 it finds designs that are very efficient (i.e. close to optimal), which means the optimisation can be stopped after just a few iterations. This is different from e.g. Ngene, which samples candidate choice tasks, and typically gradually finds better and better designs.

To see the best design that has been found, the analyst needs to press the Design tab in the right-hand side panel. When the analyst is satisfied with the design, the analyst can export the design to a text file (named "design.txt") by pushing the 'Export design' button at the top of the output panel, which will open the file in a text editor (e.g. Notepad).[5] The text file reports the design as well as additional information on the design, including the design dimension, choice probabilities and the S-estimates. Next, we provide three worked examples showing how this software could be used. Also, we interpret and discuss the resulting designs.

3.1.2 RDG Example 1: D-efficient design for estimating RRM models

In this example we create a design which is optimised for estimating a P-RRM models in MNL form, see Figure 2. The design consists of 10 choice tasks per respondent. Each choice task consists of three generic attributes: Travel Time (TT), Travel Cost (TC) and Level of Crowdedness (C). Travel Time and Travel Cost both have 6 levels; Crowdedness has two levels (not crowded or crowded). The priors for TT, TC and C are fixed values set respectively to -0.3, -2.0 and -1.6. We stopped the search algorithm after 24 iterations, which took about a minute.

[5] Note that the user requires writing access to the local Temp folder ("C:\Temp")

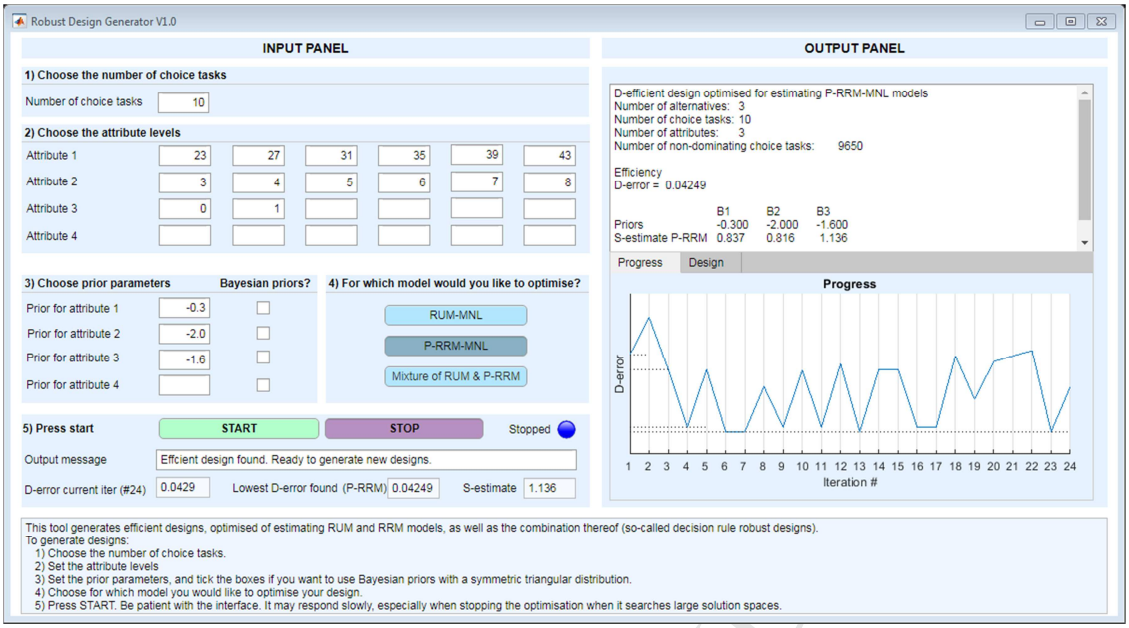


Figure 2: RDG D-efficient design for estimating RRM models, Progress tab

The best design is displayed in the output panel under the Design tab, see Figure 3. Each row corresponds to a choice task, while the columns show the attribute levels, where X_{im} denotes the attribute level for attribute *m* of alternative *i*. A number of observations can be made. Firstly, it is notable that the alternatives of the design are sorted. That is, alternative 1 is always the fastest alternative, alternative 2 is always the middle (compromise) alternative, and alternative 3 is always the slowest alternative within the choice task. The sorting facilitates understanding of the obtained design on the side of the analyst during the experimental design phase. However, when the design is used in the actual SC experiment, the analyst should shuffle the order of the alternatives and choice tasks. Secondly, we see that alternatives in the design are somewhat extreme in terms of their attribute levels. Alternative 1 and 3 almost always attain the lowest and highest levels for the travel time (respectively 23 and 43). This phenomenon is also known to occur in efficient designs optimised for RUM (Kanninen 2002). Thirdly, the S-estimates show that this design is very efficient: only two respondents are needed to obtain all parameters to be significant (i.e. given that the priors were adequately chosen). However, clearly to make inferences about a target population, a sample needs to be substantially larger than that. Fourthly, looking at the choice probabilities – which are depicted left of the design – we see that all alternatives attract substantial choice probability. This is in line with expectations based on the literature: relative to orthogonal designs efficient designs are known to generate choice tasks which are cognitively more demanding for respondents (Louviere et al. 2008), although usually not to the extent that it becomes burdensome.

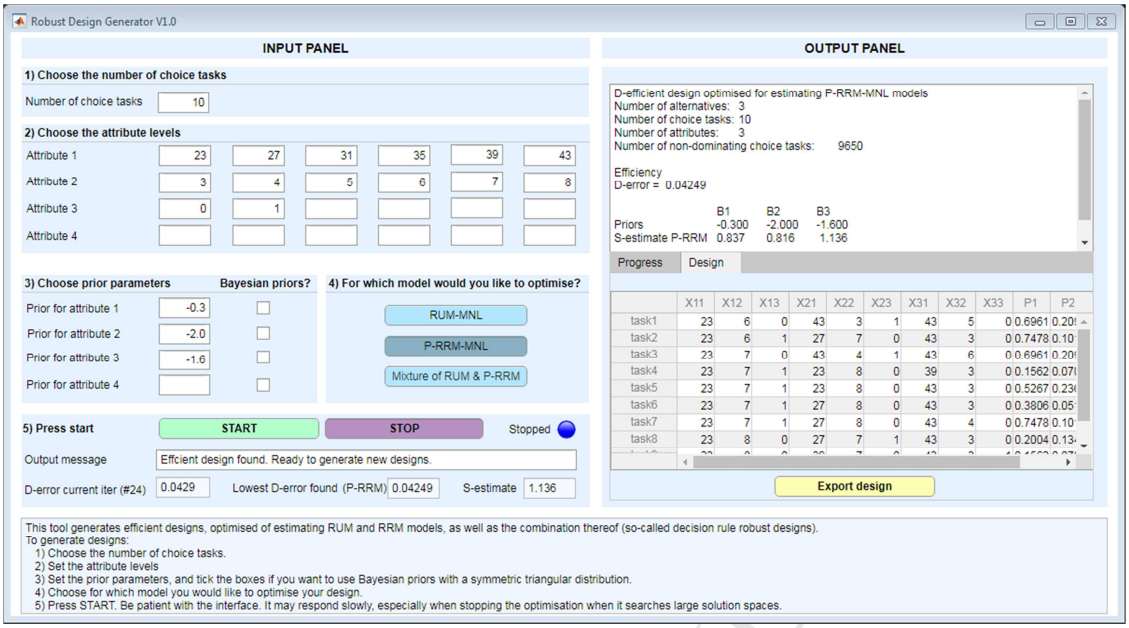


Figure 3: RDG D-efficient design for estimating RRM models, Design tab

3.1.3 RDG Example 2: D-efficient design robust for estimating RUM and RRM models

In this example we use the RDG to create a design which is optimised for the situation in which the analyst has no particular expectation regarding whether respondents are more likely to use a utility maximising or a regret minimising decision rule. Therefore, at the time of designing the experiment the analyst does not yet know which type of model he or she will estimate (RUM or RRM). Given that RUM designs can be highly inefficient for RRM, and vice versa (Van Cranenburgh et al. 2018), in this example we aim to create a design which is robust for estimating both models.

In this example we use the same design dimensions as in the previous example: 10 choice tasks per respondent, each having three alternatives, with the same attribute levels as before. To create this decision rule robust design, the 'Mixture of RUM & P-RRM' button is selected, see Figure 4. Note that both decision rules are set to contribute 'equally' to the composite D-error (i.e. $w_r = 0.5 \; \forall r$, see Equation 6). Furthermore, as shown in the output message box in Figure 4, a choice set size correction is applied to downscale the RRM model parameters towards the 'same' level as the RUM parameters, by multiplying them by 2/3 (see also section 2.2). Hence, when the 'Mixture of RUM & P-RRM' button is selected, the software considers the priors entered in the input cells to be the priors for the RUM model.

The results have two noteworthy aspects. Firstly, we see the design has somewhat more variation in the attribute levels, as compared to the design optimised for P-RRM only. For instance, while in the P-RRM efficient design the fastest alternative always costs 6, 7 or 8 euros, in the robust design in one choice task the fastest alternative costs just 5 euros. Furthermore, in only one choice task the fastest alternative costs 8 euros (the maximum level). Secondly, the composite D-error is

larger than the D-error of the design optimised for P-RRM only, see Figure 2. This is in line with expectations, given that this robust design is not solely optimised for P-RRM, but also for RUM. Finally, note that for robust designs the MNL probabilities for both RUM MNL and P-RRM MNL can be seen, by scrolling to the right.

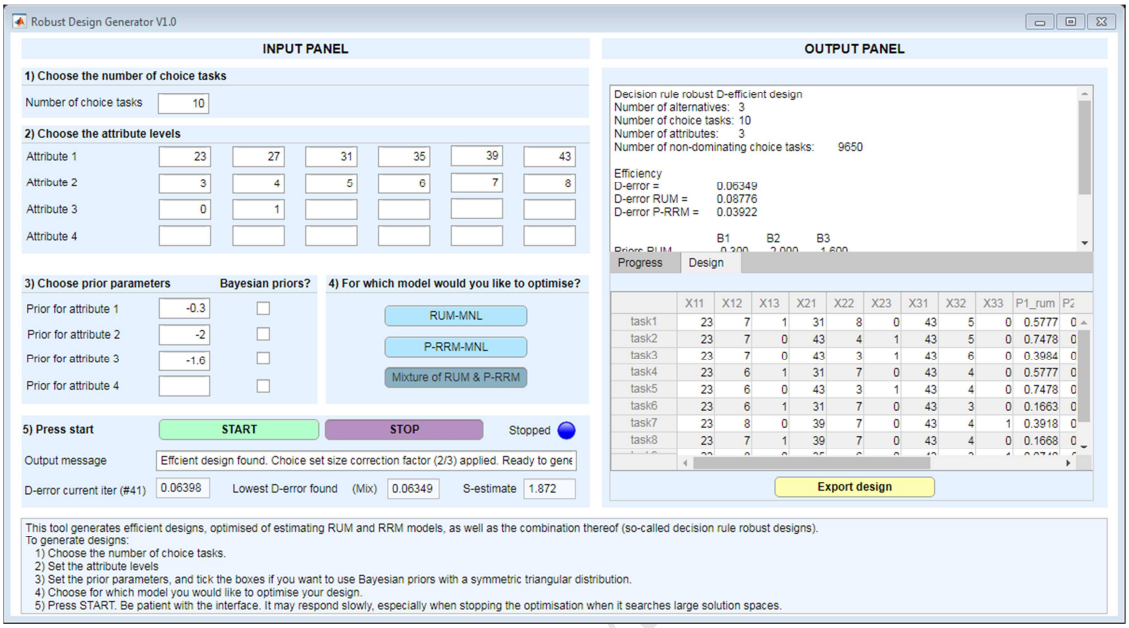


Figure 4: RDG D-efficient design robust for estimating RUM and RRM models

3.1.4 RDG Example 3: Bayesian D-efficient design robust for estimating RUM and RRM models

In this example we create a design which is robust towards two sources of uncertainty on the side of the analyst. The first type of uncertainty involves the underlying decision rule used by decision-makers (in casu: RUM or RRM); the second type of uncertainty involves the prior parameters. The analyst typically has expectations regarding the sign and sizes of the parameters, but clearly does not know their exact values. Therefore, in this example we create a design which accounts for both these sources of uncertainty. The 'Mixture of RUM & P-RRM' button is selected and the tick boxes for Bayesian priors are ticked, see Figure 5.

Figure 5 shows the Bayesian D-efficient robust design. In line with expectations, we see that the Bayesian D-error is considerably larger than in the non-Bayesian design (see Figure 4). Furthermore, at face value the design (not shown) does not look very different from the one attained in the previous example. However, in section 3.3 we will show that this design is considerably more robust towards the analyst's decision rule uncertainty than the designs in the previous two examples.

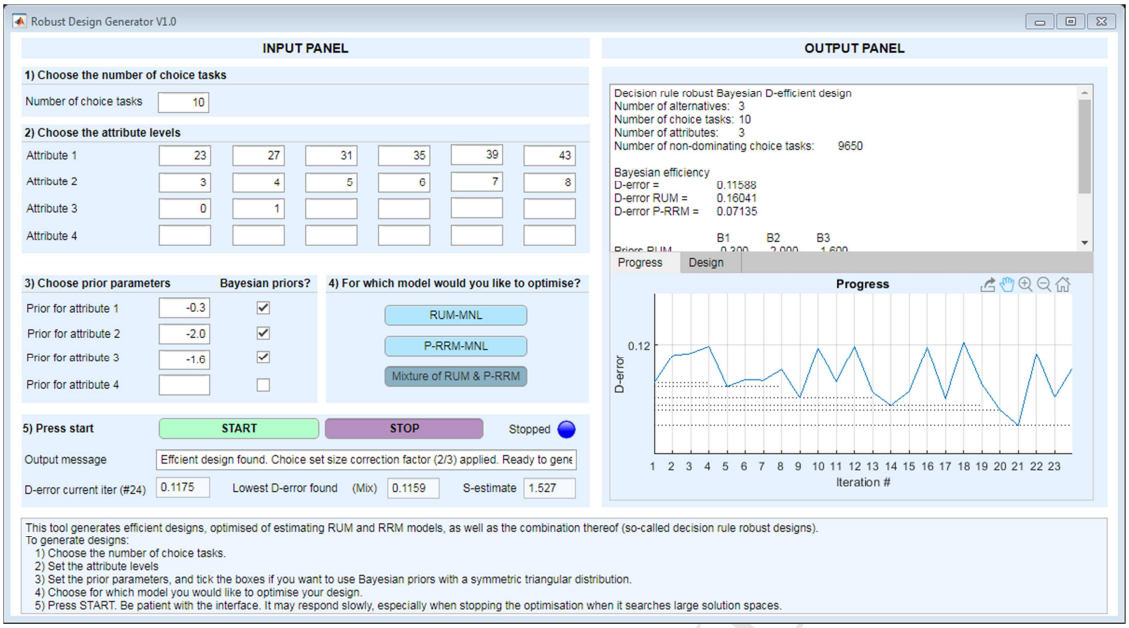


Figure 5: RDG Bayesian D-efficient design robust for RUM and RRM

3.2 Ngene software

The Ngene software package is a richly featured software tool for the generation of experimental designs for SC experiments. It is far more flexible than RDG, allowing the analyst to customise the experimental design towards his or her needs. For instance, Ngene allows the analyst to create blocked designs, pivoted designs, labelled and unlabelled designs, and constrained designs, using a range of model types, including MNL, Error Component, and Panel Mixed Logit models, and a variety of different efficiency measures, including D-error, Bayesian D-error, and S-error. Initially, efficient designs for RRM models will be released as a beta feature which is available from version Ngene 1.2.1.[6] After it has been thoroughly tested, it will be formally released in version 1.3. The Ngene software can be downloaded from www.choice-metrics.com. See this website for further details on installation and licenses.

3.2.1 Invoking RRM efficient designs in Ngene

Ngene is syntax driven, and to invoke a design optimised for estimating RRM models the `;eff` property of the `Design` command is used. The `;eff` property defines the model type together with the efficiency measure used during the optimisation. For example, to create an efficient design for the P-RRM model (with an MNL error structure), using the D-error as the efficiency measure, the analyst specifies `;eff = (mnl, d, rrm)`, where `mnl` refers to the error term assumptions of the model, `d` refers to the efficiency measure, and `rrm` refers to the assumed decision rule. Where no decision rule is explicitly inserted by the analyst, Ngene uses RUM by default. Next, we provide three examples showing how to create such designs using Ngene. The

[6] Ngene 1.2.1 was released in December 2018. This update is free-of-charge for Ngene users.

first two examples mirror the examples of the RDG tool. The third example goes beyond what can be done with the RDG tool. We discuss the designs and interpret the results in light of the results obtained using the RDG tool.

3.2.2 Ngene Example 1: D-efficient design for estimating RRM models

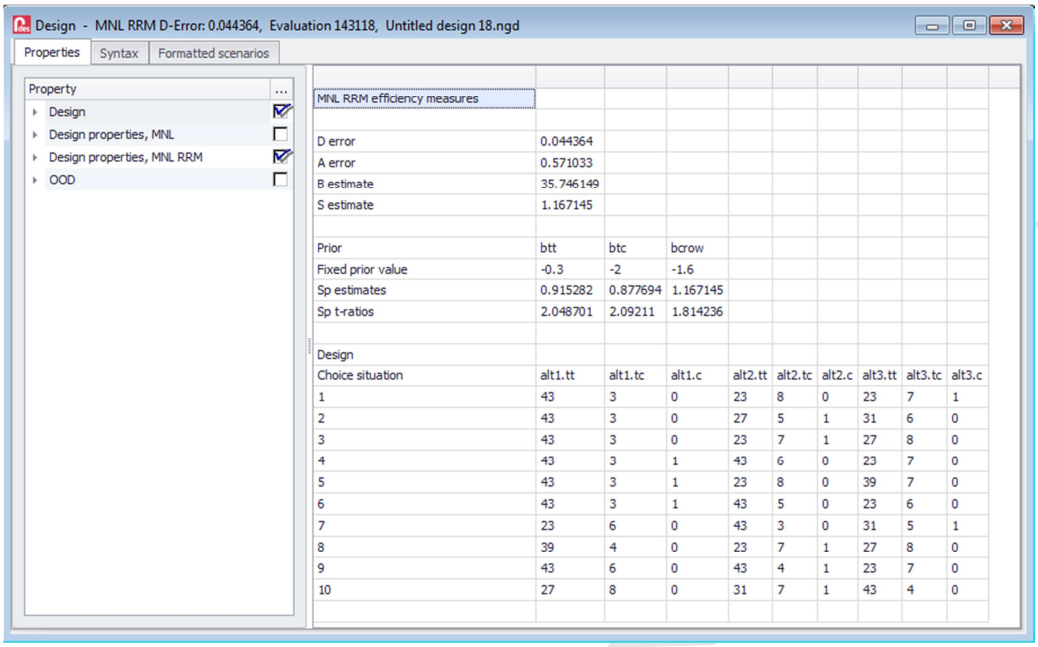
In this example we create a design which is optimised for estimating P-RRM models in the MNL form. We use the same design set-up as we used in RDG Example 1 (see section 3.1.2). This allows us to compare the outcomes of both software tools. Text box 1 shows the Ngene syntax. The `;alts` property indicates that the design is optimised for three unlabelled alternatives (`alt1`, `alt2`, `alt3`). The asterisks behind the names of the alternatives indicate that dominant alternatives are not permitted in the design. At the `;eff` property, the flags `mnl, d, rrm` indicate that the design is optimised for an P-RMM model in MNL form, using the D-error as the efficiency measure. To search the solution space a modified Fedorov search algorithm is used here.

Importantly, the analyst does not need to explicitly write down the regret functions under the `;model` property. Doing so would be laborious (due to the pair-wise nature of RRM models) as well as redundant. Rather, the analyst just needs to write down the standard linear-additive utility functions. In case the `rrm` flag is added to the `;eff` property, Ngene will automatically recognise that the design needs to be optimised for RRM and will transform the utility specification into a regret specification.

```
Design
;alts = alt1*, alt2*, alt3*
;rows = 10
;eff = (mnl,d, rrm)
;alg = mfederov
;model:

U(alt1) = btt[-0.3] * TT [23,27,31,35,39,43] + btc[-2] * TC[3,4,5,6,7,8] + bcrow[-1.6] * C[0,1]/
U(alt2) = btt       * TT                     + btc     * TC              + bcrow       * C/
U(alt3) = btt       * TT                     + btc     * TC              + bcrow       * C$
```

Text box 1: Ngene syntax for D-efficient design for estimating RRM models

Figure 6 shows the Ngene output. A number of observations can be made. Firstly, Ngene has been able to generate a design which is efficient for the P-RRM model. Looking at the D-error we see that at 0.044364 it is very close to the one found using the RDG tool (0.04249). This gives confidence that both software tools can successfully create RRM efficient designs. To further cross check the implementations in the two software tools, we evaluated the RDG design in Ngene and the Ngene design in RDG, and found identical D-errors. Secondly, although more difficult to see – since the alternatives are not sorted in Ngene – Ngene also finds a design in which the choice tasks are somewhat extreme in terms of the attribute levels. For instance, the travel time of the slowest alternative in the choice task equals 43 minutes in nine out of ten choice tasks, while the travel time of the fastest alternative in the choice task equals 23 minutes in eight out of ten choice tasks. Although both software tools find different designs, overall the designs are very similar.



MNL RRM efficiency measures

D error	0.044364		
A error	0.571033		
B estimate	35.746149		
S estimate	1.167145		
Prior	btt	btc	bcrow
Fixed prior value	-0.3	-2	-1.6
Sp estimates	0.915282	0.877694	1.167145
Sp t-ratios	2.048701	2.09211	1.814236

Design

Choice situation	alt1.tt	alt1.tc	alt1.c	alt2.tt	alt2.tc	alt2.c	alt3.tt	alt3.tc	alt3.c
1	43	3	0	23	8	0	23	7	1
2	43	3	0	27	5	1	31	6	0
3	43	3	0	23	7	1	27	8	0
4	43	3	1	43	6	0	23	7	0
5	43	3	1	23	8	0	39	7	0
6	43	3	1	43	5	0	23	6	0
7	23	6	0	43	3	0	31	5	1
8	39	4	0	23	7	1	27	8	0
9	43	6	0	43	4	1	23	7	0
10	27	8	0	31	7	1	43	4	0

Figure 6: Ngene output for D-efficient design for estimating RRM models

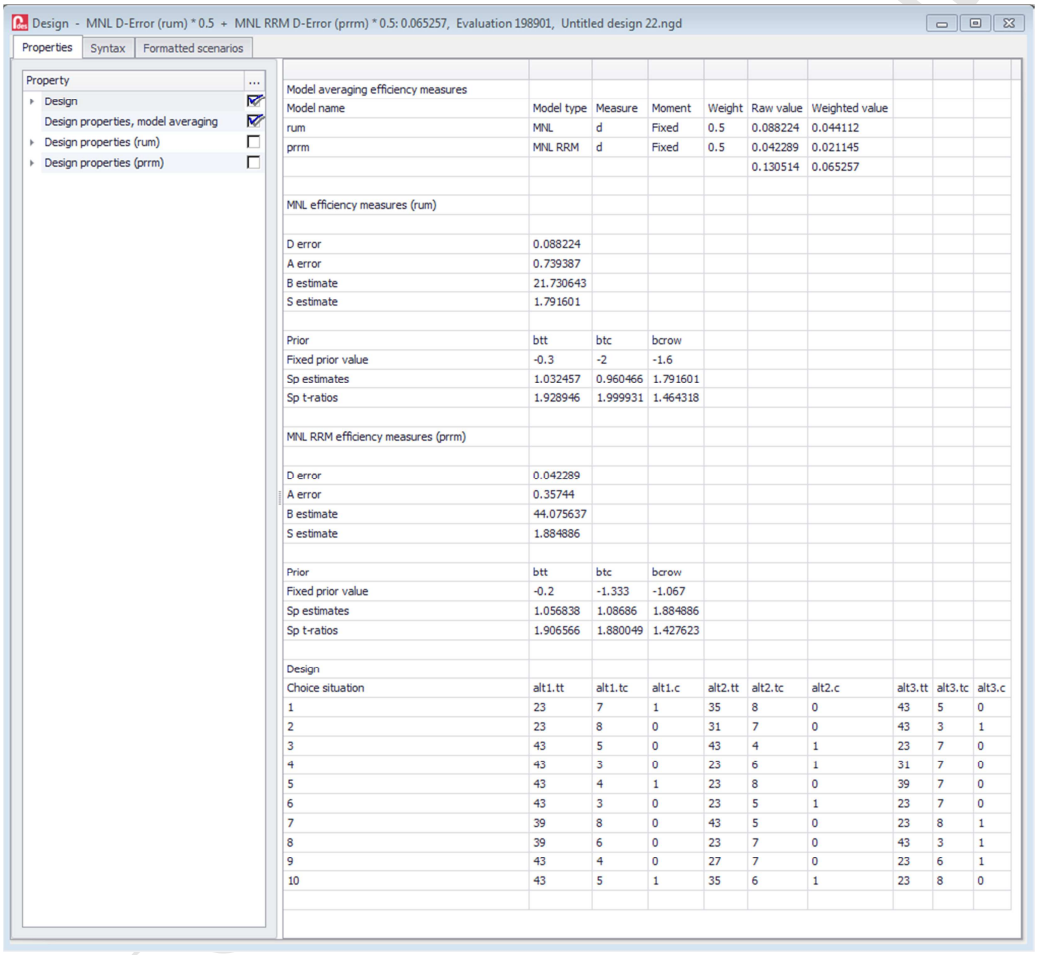
3.2.3 Ngene Example 2: D-efficient design robust for estimating RUM and RRM models

In this example we create a robust design which is efficient for estimating RUM and RRM models, using the same design set-up as in RDG Example 2 (see section 3.1.3). Text box 2 shows the Ngene syntax. To create a robust design we make use of the model averaging feature in Ngene, in two steps. Firstly, we create two models: a RUM model and a P-RRM model. Note that since Ngene automatically transforms the utility specification into a regret specification, the utility and regret functions are the same for both models (except for the prior parameters). Secondly, we compute the composite D-error using the `;eff` property. The composite D-error is the weighted sum of the RUM D-error and the P-RRM D-error. Mirroring the RDG example, in this case we use equal weighting of 0.5. Furthermore, note that the prior parameters are set to different values across the two models with the RRM priors set at 2/3 of the size of the RUM prior parameters. See Section 2.2 for setting the priors for RRM efficient designs.

```
Design
;alts(RUM)  = alt1*, alt2*, alt3*
;alts(PRRM) = alt1*, alt2*, alt3*
;rows = 10
;eff = 0.5*RUM(mnl,d,rum) + 0.5*PRRM(mnl,d,rrm)
;alg = mfederov

;model(RUM):
U(alt1) = btt[-0.3] * TT [23,27,31,35,39,43] + btc[-2]      * TC[3,4,5,6,7,8] + bcrow[-1.6]   * C[0,1]/
U(alt2) = btt       * TT                     + btc          * TC              + bcrow         * C/
U(alt3) = btt       * TT                     + btc          * TC              + bcrow         * C

;model(PRRM):
U(alt1) = btt[-0.2] * TT [23,27,31,35,39,43] + btc[-1.3333] * TC[3,4,5,6,7,8] + bcrow[-1.067] * C[0,1]/
U(alt2) = btt       * TT                     + btc          * TC              + bcrow         * C/
U(alt3) = btt       * TT                     + btc          * TC              + bcrow         * C$
```

Text box 2: Ngene syntax for D-efficient design robust for estimating RUM and RRM models

From the results in Figure 7 we see that the obtained design has a slightly worse D-error (0.065257) as compared to the D-error obtained using the RDG tool (0.06349, see section 3.1.3). However, this is most likely due to the slight difference in the prior parameter values. In the Ngene syntax we rounded off the RRM parameters to 4 decimals, while in the RDG tool they are not rounded off. Looking at the design, we see that at face value it is not very different from the RDG design. Nor does the design look very different from the one optimised for P-RRM only. However, as we will show in section 3.3 this design is considerably more robust for uncertainly regarding the underlying decision rule.



Model averaging efficiency measures

Model name	Model type	Measure	Moment	Weight	Raw value	Weighted value
rum	MNL	d	Fixed	0.5	0.088224	0.044112
prrm	MNL RRM	d	Fixed	0.5	0.042289	0.021145
					0.130514	0.065257

MNL efficiency measures (rum)

D error	0.088224
A error	0.739387
B estimate	21.730643
S estimate	1.791601

Prior	btt	btc	bcrow
Fixed prior value	-0.3	-2	-1.6
Sp estimates	1.032457	0.960466	1.791601
Sp t-ratios	1.928946	1.999931	1.464318

MNL RRM efficiency measures (prrm)

D error	0.042289
A error	0.35744
B estimate	44.075637
S estimate	1.884886

Prior	btt	btc	bcrow
Fixed prior value	-0.2	-1.333	-1.067
Sp estimates	1.056838	1.08686	1.884886
Sp t-ratios	1.906566	1.880049	1.427623

Design

Choice situation	alt1.tt	alt1.tc	alt1.c	alt2.tt	alt2.tc	alt2.c	alt3.tt	alt3.tc	alt3.c
1	23	7	1	35	8	0	43	5	0
2	23	8	0	31	7	0	43	3	1
3	43	5	0	43	4	1	23	7	0
4	43	3	0	23	6	1	31	7	0
5	43	4	1	23	8	0	39	7	0
6	43	3	0	23	5	1	23	7	0
7	39	8	0	43	5	0	23	8	1
8	39	6	0	23	7	0	43	3	1
9	43	4	0	27	7	0	23	6	1
10	43	5	1	35	6	1	23	8	0

Figure 7: Ngene output for D-efficient design robust for estimating RUM and RRM models

3.2.4 Ngene Example 3: Bayesian D-efficient design robust for estimating RUM and RRM Panel Mixed Logit models

The advantage of Ngene over RDG comes from the wide range of features that are available in Ngene. Therefore, in this final example we create a sophisticated design that takes advantage of these features, which cannot be made using the RDG tool. Nowadays, the Panel Mixed Logit model is increasingly becoming the standard model of choice for choice modellers. Therefore, in

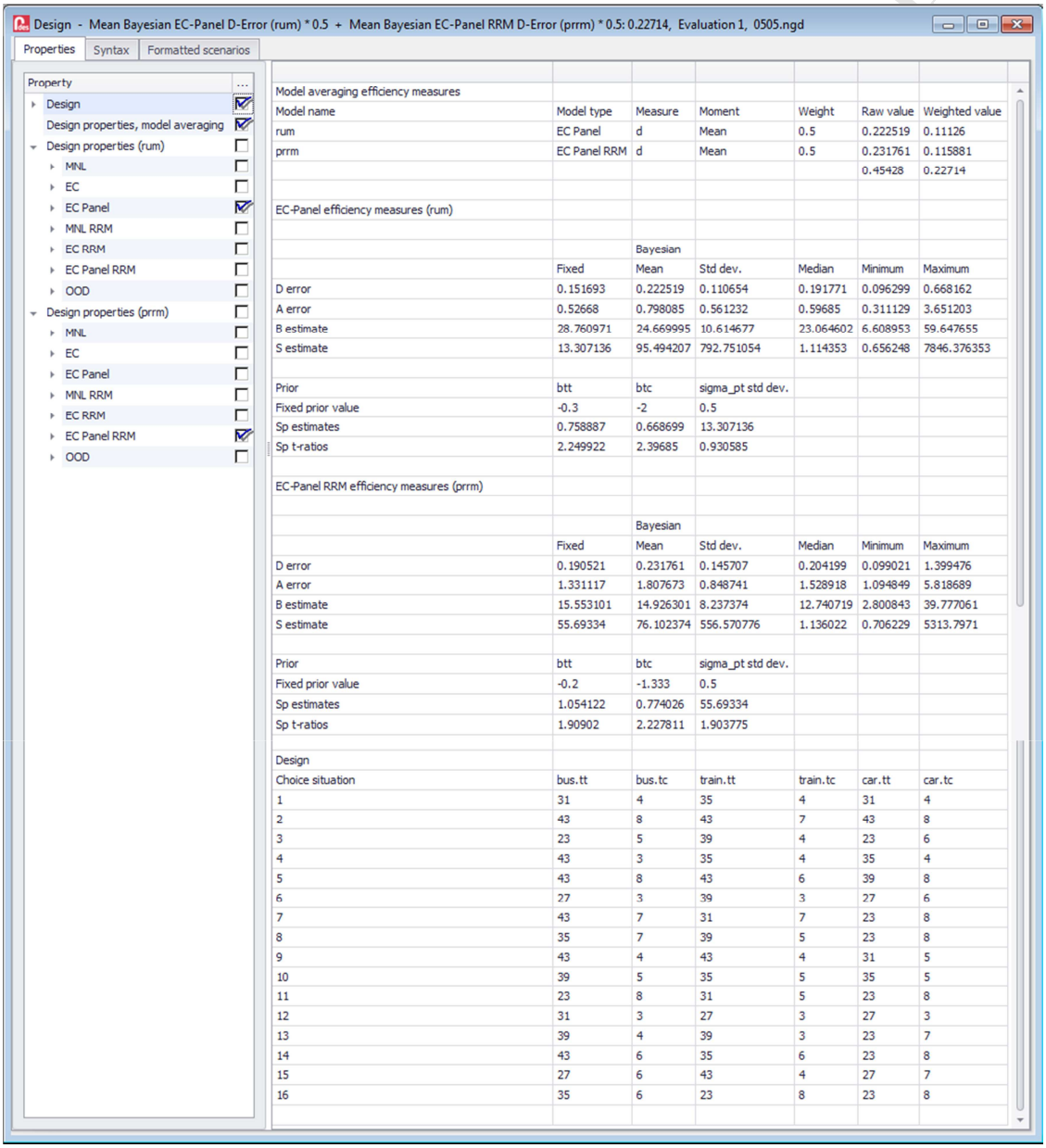
this example we create a Bayesian efficient design optimised for estimating RUM and RRM Panel Mixed Logit models. Note that generating designs which are robust for uncertainty on the side of the analyst regarding the prior parameters and are efficient for Panel Mixed Logit models is computationally very expensive as it involves taking draws for the Bayesian priors as well as draws for the random parameters.

The choice setting we look at consists of three labelled alternatives: bus, train and car, each defined by two attributes: Travel Time (TT) and a Travel Cost (TC). Since the alternatives are labelled, the utility/regret functions start with an Alternative Specific Constant (ASC), see the Ngene syntax in Text box 3. Furthermore, we account for potential correlations in the error terms caused by (1) shared unobserved utility/regret across alternatives, and (2) unobserved alternative specific variation due to the panel nature of the SC data collection. Specifically, we expect shared unobserved utility/regret across the Public Transport (PT) alternatives. Therefore, we add a normally distributed error component, `sigma_PT[ec,0.5]`, to the utility/regret functions of the PT alternatives. To account for the panel nature of the data while handling the error component, we use the `ecpanel` flag in the `;eff` property. This instructs Ngene to optimise for a Panel Mixed Logit with error components (in casu with RUM and RRM decision rules). Finally, we set a number of conditions, using the `cond` property to increase the realism of the choice tasks in the experiment. This feature allows the analyst to impose conditions on the compositions of choice tasks. In this example we impose that the car alternative is never slower and always equally or more expensive than the PT alternatives.

```
Design
;alts(RUM)  = bus, train, car
;alts(PRRM) = bus, train, car
;rows = 16
;eff =  0.5*RUM(ecpanel,d,rum,mean)    + 0.5*PRRM(ecpanel,d,rrm,mean)
;bdraws = halton(100)
;rdraws = halton(100)
;rep=100
;alg=swap

;cond:
if(bus.TT=23,car.TT=[23]),
if(bus.TT=27,car.TT=[23,27]),
if(bus.TT=31,car.TT=[23,27,31]),
if(bus.TT=35,car.TT=[23,27,31,35]),
if(bus.TT=39,car.TT=[23,27,31,35,39]),
if(train.TT=23,car.TT=[23]),
if(train.TT=27,car.TT=[23,27]),
if(train.TT=31,car.TT=[23,27,31]),
if(train.TT=35,car.TT=[23,27,31,35]),
if(train.TT=39,car.TT=[23,27,31,35,39]),
if(bus.TC=4,car.TC=[4,5,6,7,8]),
if(bus.TC=5,car.TC=[5,6,7,8]),
if(bus.TC=6,car.TC=[6,7,8]),
if(bus.TC=7,car.TC=[7,8]),
if(bus.TC=8,car.TC=[8]),
if(train.TC=4,car.TC=[4,5,6,7,8]),
if(train.TC=5,car.TC=[5,6,7,8]),
if(train.TC=6,car.TC=[6,7,8]),
if(train.TC=7,car.TC=[7,8]),
if(train.TC=8,car.TC=[8])

;model(RUM):
U(bus)  = asc_bus[-1]    + btt[(n,-0.3,0.15)] * TT [23,27,31,35,39,43] + btc[(n,-2,1)] * TC[3,4,5,6,7,8] + sigma_PT[ec,0.5]/
U(train)= asc_train[-0.5]+ btt               * TT                      + btc           * TC              + sigma_PT/
U(car)  =                  btt               * TT                      + btc           * TC

;model(PRRM):
U(bus)  = asc_bus[-1]    + btt[(n,-0.2,0.1)] * TT [23,27,31,35,39,43] + btc[(n,-1.333,0.667)] * TC[3,4,5,6,7,8]+ sigma_PT[ec,0.5]/
U(train)= asc_train[-0.5]+ btt               * TT                      + btc                   * TC             + sigma_PT/
U(car)  =                  btt               * TT                      + btc                   * TC
$
```

Text box 3: Ngene syntax for Bayesian D-efficient design robust for estimating RUM and RRM Panel Mixed Logit models

Figure 8 shows the Ngene output. It shows that using Ngene Bayesian efficient design can be created which are robust for estimating RUM and RRM Panel Mixed Logit models. The Bayesian D-errors and S-estimates show that a design has been found which is simultaneously efficient for estimating RUM and RRM Panel Mixed Logit models.



Model averaging efficiency measures						
Model name	Model type	Measure	Moment	Weight	Raw value	Weighted value
rum	EC Panel	d	Mean	0.5	0.222519	0.11126
prrm	EC Panel RRM	d	Mean	0.5	0.231761	0.115881
					0.45428	0.22714

EC-Panel efficiency measures (rum)						
		Bayesian				
	Fixed	Mean	Std dev.	Median	Minimum	Maximum
D error	0.151693	0.222519	0.110654	0.191771	0.096299	0.668162
A error	0.52668	0.798085	0.561232	0.59685	0.311129	3.651203
B estimate	28.760971	24.669995	10.614677	23.064602	6.608953	59.647655
S estimate	13.307136	95.494207	792.751054	1.114353	0.656248	7846.376353

Prior	btt	btc	sigma_pt std dev.			
Fixed prior value	-0.3	-2	0.5			
Sp estimates	0.758887	0.668699	13.307136			
Sp t-ratios	2.249922	2.39685	0.930585			

EC-Panel RRM efficiency measures (prrm)						
		Bayesian				
	Fixed	Mean	Std dev.	Median	Minimum	Maximum
D error	0.190521	0.231761	0.145707	0.204199	0.099021	1.399476
A error	1.331117	1.807673	0.848741	1.528918	1.094849	5.818689
B estimate	15.553101	14.926301	8.237374	12.740719	2.800843	39.777061
S estimate	55.69334	76.102374	556.570776	1.136022	0.706229	5313.7971

Prior	btt	btc	sigma_pt std dev.			
Fixed prior value	-0.2	-1.333	0.5			
Sp estimates	1.054122	0.774026	55.69334			
Sp t-ratios	1.90902	2.227811	1.903775			

Design						
Choice situation	bus.tt	bus.tc	train.tt	train.tc	car.tt	car.tc
1	31	4	35	4	31	4
2	43	8	43	7	43	8
3	23	5	39	4	23	6
4	43	3	35	4	35	4
5	43	8	43	6	39	8
6	27	3	39	3	27	6
7	43	7	31	7	23	8
8	35	7	39	5	23	8
9	43	4	43	4	31	5
10	39	5	35	5	35	5
11	23	8	31	5	23	8
12	31	3	27	3	27	3
13	39	4	39	3	23	7
14	43	6	35	6	23	8
15	27	6	43	4	27	7
16	35	6	23	8	23	8

Figure 8: Ngene output for Bayesian D-efficient design robust for estimating RUM and RRM Panel Mixed Logit models

3.3 Robustness of designs towards decision rule and prior misspecification

This subsection aims to analyse the robustness of the newly available designs, by adapting a prior misspecification approach used in earlier experimental design papers such as Bliemer et al. (2009) and Rose and Bliemer (2009). In subsection 3.3.1 we look at the designs generated in RDG Examples 1 to 3[7]; in subsection 3.3.2 we look at Ngene Example 3.

3.3.1 Robustness of designs from RDG Examples 1 to 3

To analyse the robustness of the newly available designs, we compute the RUM and P-RRM D-errors for a range of parameter combinations. This shows us the impacts of both decision rule and prior parameter misspecification on the efficiency of the design. More specifically, we vary the cost and time parameters, while keeping the crowdedness parameter constant at the true level (β_{crow} = -1.6). To assess the robustness as cleanly as possible, we compute the D-errors by evaluating the part of the AVC for the cost and time parameters only (i.e. the parameter for crowdedness is not considered in the computed D-errors). In the literature, this is known as the D_S-efficiency (Atkinson et al. 2007). The range of the cost and time parameters are chosen symmetrically around the prior parameters which are used for optimising the designs (i.e. β_{tt} = -0.30 with a range from -0.1 to -0.5, β_{tc} = -2.0 with a range from -0.5 to -3.5).

Figure 9 shows the RUM and P-RRM D-errors as a function of the combination of the cost and time parameters, conditional on the P-RRM optimised design (RDG Example 1, see section 3.1.2). The left-hand side plot shows the D-errors for in case a P-RRM MNL model is estimated using this design; the right-hand side plot shows the D-errors in case a RUM MNL model is estimated using this design. Based on Figure 9 we can draw a number of conclusions. Firstly, the left-hand plot shows that the P-RRM design is efficient for estimating P-RRM models, given that the ratio of the parameters are not too much outside the range for which the trade-offs are optimised, which is around $\beta_{tt}/\beta_{tc} = 0.15$. The design becomes considerably less efficient in case $\beta_{tt}/\beta_{tc} < 0.10$ (upper left corner) or $\beta_{tt}/\beta_{tc} > 0.25$ (lower right corner). Secondly, the right-hand side plot shows that although the design is optimised for the P-RRM model, it is reasonably efficient for estimating RUM MNL models, given that the parameters are within the blue coloured band. However, a particular concern here is that the blue coloured band is off the diagonal. Therefore, when in this case the true parameters happen to be considerably larger than the analyst expected when creating the experimental design, the analyst may find he has used an inefficient design, even though he set the ratios of the parameters well. This result supports the notion that efficient designs optimised for one particular decision rule can be inefficient for estimating models having another decision rule.

[7] We have also conducted these analyses using the designs from Ngene Examples 1 and 2, and found virtually identical results. Therefore, for reasons of brevity we do not show them here. However, an overview of the statistics of all the created designs can be found in Appendix A.

Figure 10 shows the RUM and P-RRM D-errors, conditional on the decision rule robust design (RDG Example 2, see section 3.1.3). In comparison with Figure 9 we see two key differences. Firstly, at the left-hand side plot the width of the blue coloured band where the design is efficient has substantially increased. Therefore, this design is more robust towards prior parameter misspecification. Secondly, and more crucially, the right-hand side plot shows that this design is much better suited for estimating RUM models than the P-RRM optimised design shown Figure 9. In this design the blue coloured band neatly follows the diagonal, meaning that as long at the parameter ratios are fairly well chosen this design is efficient for estimating both RUM and RRM models. This result provides further evidence that creating designs that are robust for multiple decision rules is sensible when the analyst does not know up front which models (including their embedded decision rules) he or she will estimate after having collected the data.

Finally, Figure 11 shows the P-RRM and RUM D-errors for the Bayesian efficient decision rule robust design (RDG Example 3, see section 3.1.4). In line with expectations, it shows that this design is efficient for a wide range of parameter values, both for estimating RUM and P-RRM models. The fact that both the right and left-hand side plots are overall blue shows that designs can be generated which are robust towards uncertainty regarding the underlying decision rule and regarding the values of the model parameters at the same time, at fairly limited statistical cost. Therefore, in case the analyst is uncertain about the models, including their embedded decision rules, he or she will estimate using the data, as well as the size and ratios of the model parameters, it is sensible to use Bayesian efficient designs which are robust towards multiple decision rules (in casu RUM and RRM).

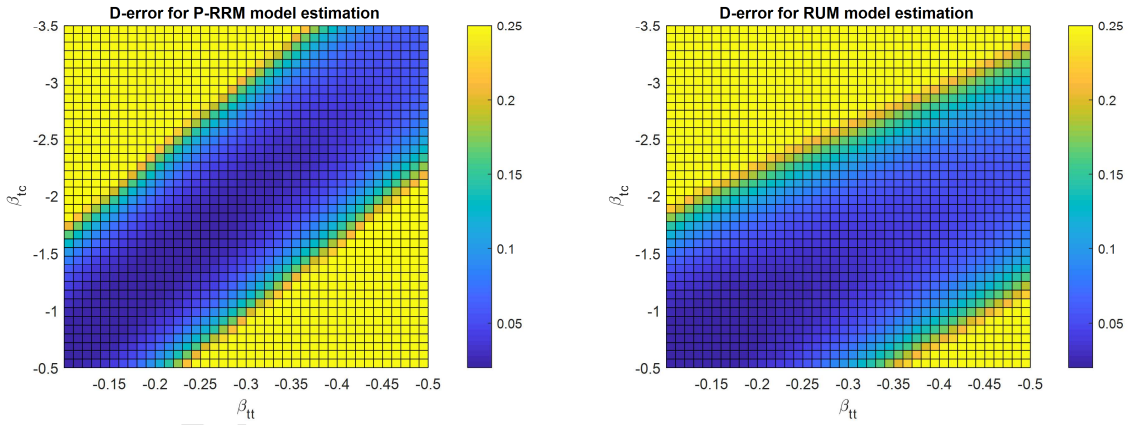


Figure 9: D-error for P-RRM optimised design as a function of β_{tt} and β_{tc}

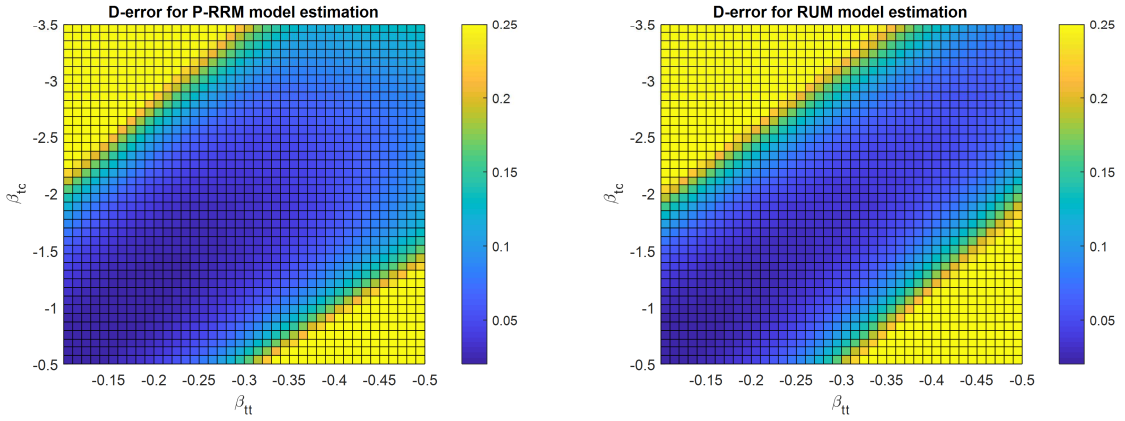


Figure 10: D-error for decision rule robust design as a function of β_{tt} and β_{tc}

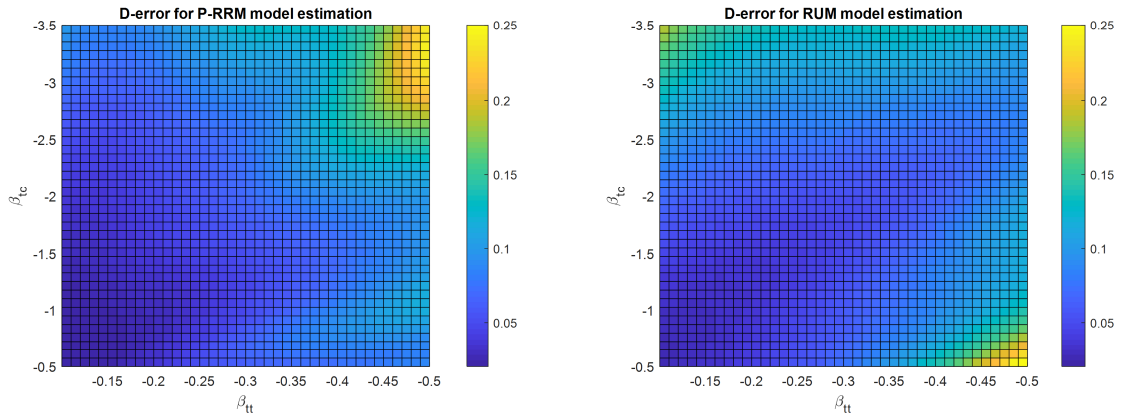


Figure 11: D-error for decision rule robust Bayesian design as a function of β_{tt} and β_{tc}

3.3.2 Robustness of design from Ngene Example 3

In section 3.2.4 we showed that Bayesian efficient designs robust for estimating RUM and RRM Panel Mixed Logit models can be created using Ngene. However, it does not immediately follow that creating such decision rule robust designs is worthwhile in this context (especially considering that creating such designs is computationally heavy). Hence, one could question whether a Bayesian efficient design optimised for estimating a RUM Panel Mixed Logit model would work (almost) as well, in terms of its robustness towards the two different decision rules.

To answer this question, ideally we would conduct the same analysis as in the previous subsection as this would give us insights on the robustness towards decision rule and prior parameter misspecification. However, conducting that sort of analysis is computationally infeasible. Therefore, in this sub section we look at the robustness towards decision rule misspecification only. To do so, we have created seven Bayesian D-efficient design specifications robust for estimating RUM and RRM Panel Mixed Logit models, using the syntax in Text box 3. But, here we varied the decision rule weights, w_{RUM} and w_{RRM} . Specifically, we let w_{RUM} range

from 0 to 1 in six intervals, while imposing that the sum of the weights equals 1 (i.e., $w_{RUM} + w_{RRM} = 1$). Each design was optimised during a period of 12 full hours. Furthermore, to avoid presenting results that are due to a particular manifestation of random draws or starting designs we ran each code to generate the designs fifteen times.

Figure 12 shows the results. The left-hand side plot and middle plot show respectively the RUM Bayesian D-efficiency and the P-RRM Bayesian D-efficiency of the 7 × 15 = 105 designs, as a function of the decision rule weights: w_{RUM} at the bottom of the *x*-axis, w_{RRM} at the top. The key insight these two plots provide is that also for complex Bayesian efficient designs, like this one, misspecification of the decision rule can have considerable impact on the statistical efficiency. This can readily be seen by looking at the differences in the D-errors between the outer left ($w_{RUM} = 0$) and right ($w_{RUM} = 1$) side of each plot. Specifically, the left-hand side plot shows that for estimating a RUM model the design optimised for RUM attains (on average) a D-error of 0.229, while a design optimised for P-RRM attains (on average) a D-error of 0.274. Hence, a full P-RRM design is almost 20% (0.274/0.229 = 1.20) less efficient to estimate a RUM model than a design optimised for RUM only. Likewise, the middle plot shows that for estimating a P-RRM model the design optimised for P-RRM attains (on average) a D-error of 0.218, while a design optimised for RUM attains (on average) a D-error of 0.295. This implies that a full RUM design is on average about 37% (0.295/0.215 = 1.37) less statistically efficient to estimate the P-RRM model than a design optimised for P-RRM only.

In the right-hand side plot of Figure 12, the RUM and P-RRM Bayesian D-efficiency are scattered against each other. This plot shows that also in the context of complex Bayesian designs, the use of a decision rule robust design is a sound strategy. Specifically, it shows that designs that are optimised for both RUM and P-RRM (depicted in yellow) are doing relatively well across the board. To highlight this we grouped the designs into three categories: (1) *RUM optimised designs* (green), where $w_{RUM} = \{1, 0.9\}$, (2) *Decision rule robust designs* (yellow), where $w_{RUM} = \{0.7, 0.5, 0.3\}$, and *P-RRM optimised designs* (purple), where $w_{RUM} = \{0.1, 0\}$. Then, for each category a convex hull is plotted. The boundaries of the convex hulls show that the decision rule robust designs (yellow) are at the frontier. That is, they are located more towards the lower left corner than the other two categories. For instance, a particularly good design is the one in the lower left corner: {0.228, 0.217}. This design is just 3% (0.228/0.222 = 1.03) less efficient than the most efficient RUM optimised designs (green), and just 7% less efficient (0.217/0.203 = 1.07) than the most efficient P-RRM optimised designs (purple). This shows that decision rule robust designs can come at fairly limited statistical cost.

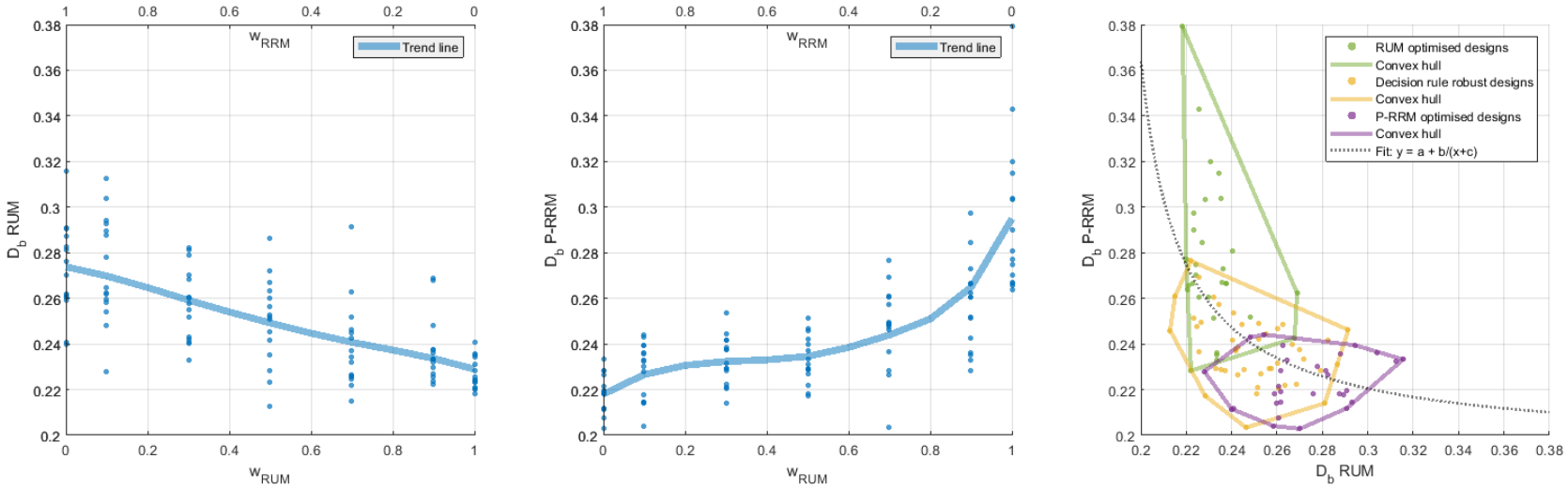


Figure 12: Effect of the decision rule weights

4 Conclusions and discussion

In this paper we have presented two new software tools that can be used to generate SC experimental designs that are simultaneously efficient for estimating RUM and RRM models. Therefore, these designs are robust towards uncertainty on the side of the analyst – at the time of creating the experimental design – regarding the underlying decision rule (in casu: RUM and RRM). To facilitate creating such decision rule robust designs, in this paper we have discussed practical issues encountered when generating such designs, such as how to obtain priors for RRM models, and presented worked examples. Finally, we have analysed the robustness of the designs that we have created using the newly available software tools. Our results provide new evidence that designs optimised for one decision rule can be inefficient for another. Earlier studies have shown this using local (i.e. non-Bayesian) designs. Our results indicate this also holds for Bayesian efficient designs. These results highlight the added value of decision rule robust designs.

The new software tools presented in this paper enable scrutinisation of outstanding and new research questions in the fields of experimental design and choice modelling. A first outstanding and in our view highly relevant empirical question that can be scrutinised is whether certain types of designs increase the prevalence of certain decision rules. It is clear that designs can be created which trigger a specific behavioural phenomena, such as the compromise effect, see for instance Guevara and Fukushi (2016). However, at present it is not clear whether designs that are *statistically* optimised for one particular decision rule also trigger that decision rule on the side of the respondent (while keeping the design dimensions constant). Van Cranenburgh et al. (2018) report some evidence for this. On their data, they find that the RUM model obtains the best model fit when a RUM optimised design is presented to the respondents, while the P-RRM model obtains the best model fit when a P-RRM optimised design is presented (to the same group of respondents). In case more empirical evidence is found that designs statistically optimised for RUM increase the prevalence of a RUM decision rule on the side of the respondents, this would shed new light on the existing literature on decision rules based on SP data.

A second outstanding research question that can be scrutinised using the new software tools concerns the statistical efficiency of P-RRM efficient designs for other types of RRM and non-RUM models, such as the G-RRM, μRRM, and RAM model (Chorus 2014; Leong and Hensher 2015; Van Cranenburgh et al. 2015a). Conceptually, it seems intuitive that a design which is efficient for P-RRM is also fairly efficient for other types of RRM models. However, currently this is not established as a fact. We believe further investigation is warranted.

A third research question relates to the best way to create designs which are jointly optimised for multiple decision rules. In this paper, we have used a relatively straightforward averaging approach, in which equal weights w_r are given to the decision rules. However, their actual contributions to the composite D-error have not been equal, due to the fact that the D-errors are not equally large. Future research may explore the impact of this on the robustness, as well as find methods to account for it, for example by means of iteratively rescaling the weights.

Acknowledgements

The authors would like to thank Prof. Michiel Bliemer, Dr. Marco Kouwenhoven and Prof. Caspar Chorus for their valuable contributions to the development of the software tools.

Appendix A: Overview of the statistics of designs

Model to estimate	D-errors			S-estimates					
	RUM	P-RRM	RUM-P-RRM	RUM			P-RRM		
				β_{tt}	β_{tc}	β_{crow}	β_{tt}	β_{tc}	β_{crow}
RDG designs									
1 P-RRM	N/A	0.043	N/A	N/A	N/A	N/A	0.837	0.816	1.136
2 Robust design RUM-P-RRM	0.088	0.039	0.064	0.975	0.931	1.872	0.462	0.460	0.812
3 Bayesian robust design RUM-P-RRM	0.167[I]	0.072[I]	0.119[I]	0.928	0.852	1.609	0.505	0.489	0.757
Ngene designs									
1 P-RRM	N/A	0.046	N/A	N/A	N/A	N/A	0.920	0.882	1.116
2 Robust design RUM-P-RRM	0.088	0.042	0.065	1.032	0.960	1.792	1.057	1.087	1.885
3 Bayesian D-efficient design robust design for Panel Mixed Logit model	0.223[I]	0.232[I]	0.227[I]	0.759	0.669	N/A	1.054	0.774	N/A

[I] Bayesian D-error

Appendix B: Alternative Specific Constants in P-RRM models

For RRM models that have a logarithmic attribute level regret function, it has been noticed in the literature that RUM and RRM treatment of ASCs, and other variables which are binary in the difference across alternatives, is mathematically equivalent, apart from a rescaling which has no impact on the statistical efficiency or model fit (Chorus 2012; Hess et al. 2014). In this appendix we extent this result, and show that RUM and P-RRM treatment of such variables is mathematically fully equivalent, i.e. without a rescaling.

Proof
If RUM and P-RRM treatment of variables which are binary in the difference across alternatives is mathematically equivalent, then RUM and P-RRM should generate the same choice probabilities in choice tasks comprising of such variables only. In discrete choice models the absolute level of utility is irrelevant, and only utility differences matter. Therefore, it should hold that $V_j - V_i = -\left(R_j - R_i\right)$.

For reasons of exposition, let's consider the simple situation in which there are three alternatives consisting of two variables which are binary in the difference across alternatives (Table B1).

	Alternative 1	**Alternative 2**	**Alternative 3**
Attr. 1	0	1	0
Attr. 2	0	0	1

RUM model
In case the DGP is linear-additive RUM, then the observed utilities are:

$$V_1 = 0, V_2 = \beta_1, V_3 = \beta_2$$

The utility differences are given by:

$$V_2 - V_1 = \beta_1,\ V_3 - V_1 = \beta_2$$

P-RRM model
In case the DGP is P-RRM, then the observed regrets are:

$$R_1 = \max\left(0, \theta_1 \cdot [1-0]\right) + \max\left(0, \theta_1 \cdot [0-0]\right) + \max\left(0, \theta_2 \cdot [0-0]\right) + \max\left(0, \theta_2 \cdot [1-0]\right) = \theta_1 + \theta_2$$
$$R_2 = \max\left(0, \theta_1 \cdot [0-1]\right) + \max\left(0, \theta_1 \cdot [0-1]\right) + \max\left(0, \theta_2 \cdot [0-0]\right) + \max\left(0, \theta_2 \cdot [1-0]\right) = \theta_2$$
$$R_3 = \max\left(0, \theta_1 \cdot [0-0]\right) + \max\left(0, \theta_1 \cdot [1-0]\right) + \max\left(0, \theta_2 \cdot [0-1]\right) + \max\left(0, \theta_2 \cdot [0-1]\right) = \theta_1$$

The regret differences are given by:

$$\left(R_2 - R_1\right) = \left(\theta_2 - \left(\theta_1 + \theta_2\right)\right) = -\theta_1$$
$$\left(R_3 - R_1\right) = \left(\theta_1 - \left(\theta_1 + \theta_2\right)\right) = -\theta_2$$

Test of equivalence
Equating utility differences to regret difference, yields:

$$V_2 - V_1 = -\left(R_2 - R_1\right) \quad \rightarrow \beta_1 = \theta_1$$
$$V_3 - V_1 = -\left(R_3 - R_1\right) \quad \rightarrow \beta_2 = \theta_2$$

This result shows that RUM and P-RRM treatment of variables which are binary in the difference across alternatives is mathematically equivalent. Hence, an analyst will obtain exactly the same model fit and parameter estimates, regardless of whether he or she treats the binary in the difference variables as RUM or P-RRM. Note that by extension, it is impossible for an analyst to tell which decision rule (RUM or P-RRM) better explains the observed choices, based on data consisting of binary in the difference variables only. Finally, it can easily be seen that this proof generalises towards more than three alternatives and more than two binary in the difference variables.

Q.E.D.

References

Atkinson, A., Donev, A. & Tobias, R. (2007). *Optimum experimental designs, with SAS*: Oxford University Press).

Bliemer, M. C. & Collins, A. T. (2016). On determining priors for the generation of efficient stated choice experimental designs. *Journal of Choice Modelling, 21*, 10-14.

Bliemer, M. C., Rose, J. M. & Hensher, D. A. (2009). Efficient stated choice experiments for estimating nested logit models. *Transportation Research Part B: Methodological, 43(1)*, 19-35.

Bliemer, M. C. J. & Rose, J. M. (2011). Experimental design influences on stated choice outputs: An empirical study in air travel choice. *Transportation Research Part A: Policy and Practice, 45(1)*, 63-79.

ChoiceMetrics (2018) *Ngene 1.2 User Manual & Reference Guide*. Sydney, Australia.

Chorus, C. (2012). Random Regret Minimization: An Overview of Model Properties and Empirical Evidence. *Transport Reviews, 32(1)*, 75-92.

Chorus, C. G. (2010). A new model of random regret minimization. *European Journal of Transport and Infrastructure Research, 10(2)*, 181-196.

Chorus, C. G. (2014). A Generalized Random Regret Minimization model. *Transportation Research Part B: Methodological, 68(0)*, 224-238.

Chorus, C. G. & Bierlaire, M. (2013). An empirical comparison of travel choice models that capture preferences for compromise alternatives. *Transportation, 40(3)*, 549-562.

de Bekker-Grob, E. W., Ryan, M. & Gerard, K. (2012). Discrete choice experiments in health economics: a review of the literature. *Health economics, 21(2)*, 145-172.

Fedorov, V. V. (1972). *Theory of optimal experiments*: Elsevier).

Ferrini, S. & Scarpa, R. (2007). Designs with a priori information for nonmarket valuation with choice experiments: A Monte Carlo study. *Journal of Environmental Economics and Management, 53(3)*, 342-363.

Guevara, C. A. & Fukushi, M. (2016). Modeling the decoy effect with context-RUM Models: Diagrammatic analysis and empirical evidence from route choice SP and mode choice RP case studies. *Transportation Research Part B: Methodological, 93, Part A*, 318-337.

Hancock, T. O., Hess, S. & Choudhury, C. F. (2018). Decision field theory: Improvements to current methodology and comparisons with standard choice modelling techniques. *Transportation Research Part B: Methodological, 107*, 18-40.

Hess, S., Beck, M. J. & Chorus, C. G. (2014). Contrasts between utility maximisation and regret minimisation in the presence of opt out alternatives. *Transportation Research Part A: Policy and Practice, 66(0)*, 1-12.

Hess, S., Stathopoulos, A. & Daly, A. (2012). Allowing for heterogeneous decision rules in discrete choice models: an approach and four case studies. *Transportation, 39(3)*, 565-591.

Huber, J. & Zwerina, K. (1996). The Importance of Utility Balance in Efficient Choice Designs. *Journal of Marketing Research, 33(3)*, 307-317.

Kanninen, B. J. (2002). Optimal Design for Multinomial Choice Experiments. *Journal of Marketing Research, 39(2)*, 214-227.

Kessels, R., Goos, P. & Vandebroek, M. (2006). A Comparison of Criteria to Design Efficient Choice Experiments. *Journal of Marketing Research, 43(3)*, 409-419.

Kessels, R., Jones, B., Goos, P. & Vandebroek, M. (2008). Recommendations on the use of Bayesian optimal designs for choice experiments. *Quality and Reliability Engineering International, 24(6)*, 737-744.

Kessels, R., Jones, B., Goos, P. & Vandebroek, M. (2011). The usefulness of Bayesian optimal designs for discrete choice experiments. *Applied Stochastic Models in Business and Industry, 27(3)*, 173-188.

Kivetz, R., Netzer, O. & Srinivasan, V. (2004). Alternative Models for Capturing the Compromise Effect. *Journal of Marketing Research, 41(3)*, 237-257.

Leong, W. & Hensher, D. A. (2012). Embedding Decision Heuristics in Discrete Choice Models: A Review. *Transport Reviews, 32(3)*, 313-331.

Leong, W. & Hensher, D. A. (2015). Contrasts of relative advantage maximisation with random utility maximisation and regret minimisation. *Journal of Transport Economics and Policy (JTEP), 49(1)*, 167-186.

Louviere, J., xa, J, Islam, T., Wasi, N., Street, D., Burgess, L., John Deighton served as, e. & Tulin Erdem served as associate editor for this, a. (2008). Designing Discrete Choice Experiments: Do Optimal Designs Come at a Price? *Journal of Consumer Research, 35(2)*, 360-375.

Louviere, J. J., Hensher, D. A. & Swait, J. D. (2000). *Stated choice methods : analysis and applications*. (Cambridge, UK: Cambridge University Press).

Rose, J. M. & Bliemer, M. C. J. (2009). Constructing Efficient Stated Choice Experimental Designs. *Transport Reviews: A Transnational Transdisciplinary Journal, 29(5)*, 587-617.

Rose, J. M. & Bliemer, M. C. J. (2013). Sample size requirements for stated choice experiments. *Transportation, 40(5)*, 1021-1041.

Rose, J. M., Scarpa, R. & Bliemer, M. C. (2009). *Incorporating model uncertainty into the generation of efficient stated choice experiments: A model averaging approach*: Institute of Transport and Logistics Studies).

Sandor, Z. & Wedel, M. (2001). Designing conjoint choice experiments using managers' prior beliefs. *Journal of Marketing Research, 38(4)*, 430-444.

Street, D. J. & Burgess, L. (2007). *The construction of optimal stated choice experiments: Theory and methods*: John Wiley & Sons).

Van Cranenburgh, S. & Alwosheel, A. (2019). An artificial neural network based approach to investigate travellers' decision rules. *Transportation Research Part C: Emerging Technologies, 98*, 152-166.

Van Cranenburgh, S. & Bliemer, M. C. J. (2018). Information theoretic-based sampling of observations. *Journal of Choice Modelling*.

Van Cranenburgh, S. & Chorus, C. G. (2018). Does the decision rule matter for large-scale transport models? *Transportation Research Part A: Policy and Practice, 114*, 338-353.

Van Cranenburgh, S., Guevara, C. A. & Chorus, C. G. (2015a). New insights on random regret minimization models. *Transportation Research Part A: Policy and Practice, 74(0)*, 91-109.

Van Cranenburgh, S. & Prato, C. G. (2016). On the robustness of Random Regret Minimization modelling outcomes towards omitted attributes. *Journal of Choice Modelling,(18)*, 51-70.

Van Cranenburgh, S., Prato, C. G. & Chorus, C. (2015b) Accounting for variation in choice set size in Random Regret Minimization models,

Van Cranenburgh, S., Rose, J. M. & Chorus, C. G. (2018). On the robustness of efficient experimental designs towards the underlying decision rule. *Transportation Research Part A: Policy and Practice, 109*, 50-64.

Walker, J. L., Wang, Y., Thorhauge, M. & Ben-Akiva, M. (2018). D-efficient or deficient? A robustness analysis of stated choice experimental designs. *Theory and Decision, 84(2)*, 215-238.

Statement of contribution

Stated Choice (SC) experiments are widely used to acquire understanding of choice behaviour in a variety of research fields, including but not limited to transportation, marketing, and health and environmental economics. Software to create efficient experimental designs for SC experiments have exclusively been based on the (often implicit) assumption that decision-makers make choices using a (linear-additive) Random Utility Maximisation (RUM) decision rule. However, a growing number of studies have found overwhelming evidence that decision-makers may opt for other types of decision rules when making choices. In light of this, recently a method to create efficient experimental designs for one alternative decision rule, namely Random Regret Minimisation (RRM), has been proposed. This development opens up the possibility to create designs that are simultaneously efficient for both decision rules (RUM and RRM). However, although the theory to devise such decision rule robust designs has been established, the burden to actually create them is currently high: it requires extensive software coding on the side of the analyst.

The contribution of this work is that it lowers the burden for analysts who wish to create experimental designs that are simultaneously efficient for estimating RUM and RRM models. In particular, it presents two software tools in which such *decision rule robust designs* can be created. The first software tool –called Robust Design Generator (RDG)– is a lean, easy-to-use and free-of-charge experimental design tool. This tool is confined to the design of unlabelled experiments with three alternatives. The second tool constitutes a newly developed extension of Ngene. Ngene is an established, highly versatile software package dedicated to the design of SC experiments. To facilitate the use of the new software tools, this paper presents worked examples and focusses on practical issues encountered when generating such decision rule robust designs.

Presents two software tools for creating decision rule robust experimental designs

Software tool 1 -called RDG- is lean, easy-to-use and free-of-charge

Software tool 2 constitutes an Ngene extension and provides full user flexibility

We show that decision rule robust designs are efficient for both RUM and RRM

We show that designs optimised for RUM can be inefficient for RRM, and vice versa