Journal of Statistical Software
October 2016, Volume 74, Issue 3. doi: 10.18637/jss.v074.i03

Flexible Graphical Assessment of Experimental Designs in R: The vdg Package

Pieter C. Schoonees
Rotterdam School of Management,
Erasmus University

Niël J. Le Roux
University of
Stellenbosch

Roelof L. J. Coetzer
Group Technology,
Sasol South Africa

Abstract

The R package **vdg** provides a flexible interface for producing various graphical summaries of the prediction variance associated with specific linear model specifications and experimental designs. These methods include variance dispersion graphs, fraction of design space plots and quantile plots which can assist in choosing between a catalogue of candidate experimental designs. Instead of restrictive optimization methods used in traditional software to explore design regions, **vdg** utilizes sampling methods to introduce more flexibility. The package takes advantage of R's modern graphical abilities via **ggplot2** (Wickham 2009), adds facilities for using a variety of distance methods, allows for more flexible model specifications and incorporates quantile regressions to help with model comparison.

Keywords: experimental design, fraction of design space plot, variance dispersion graph, scaled prediction variance.

1. Introduction

A challenging problem in experimental design is that of choosing the most appropriate design from a catalogue of possibilities, given a specific model form. The designs available will most likely originate from either traditional design theory, such as well-known (fractional) factorial or Box-Behnken designs, or as algorithmically constructed designs according to optimal design theory. Optimal designs have a more tailor-made character and are constructed by choosing the design points so that the selected design criterion is optimized. Such optimality criteria are usually referred to by a letter of the alphabet, such as in D- or I-optimality for "determinant" and "integrated variance" optimality respectively (see Atkinson, Donev, and Tobias 2007, for further details).

Whereas traditional experimental design theory depends mostly on qualitative aspects of designs to choose the most suitable design, comparisons based on the value of the optimality criterion are comparatively simple and convenient. This is often justifiable because many standard designs are also optimal for specific models and design criteria, but in nonstandard design settings this method can be dangerous. Since an optimality criterion is a single number summary of a multidimensional concept, it supplies limited information. For example, the D-criterion minimizes the generalized variance of the parameter estimates for a linear regression model but conveys nothing about the estimation accuracy of the individual parameters.

Consequently, graphical procedures for design assessment have been developed to enable researchers to study designs more thoroughly. The main types of procedures are the variance dispersion graph (VDG; Giovannitti-Jensen and Myers 1989), the related quantile plot (QP; Khuri, Kim, and Um 1996) and the more recent fraction of design space plot (FDS; Zahran, Anderson-Cook, and Myers 2003). These graphs provide a less contrived summary of the prediction variance properties of an experimental design for a given model. The prediction variance is especially important where models are used for process optimization, such as in response surface methodology (RSM; Myers, Montgomery, and Anderson-Cook 2009).

There are currently two similar packages available for R (R Core Team 2016) on the Comprehensive R Archive Network (CRAN), namely package **Vdgraph** (Lawson 2014) and package **VdgRsm** (Srisuradetchai and Borkowski 2015). **Vdgraph** provides a front-end for the original Fortran program of Vining (1993a,b) for constructing VDGs, rewritten and simplified to a subroutine. This package allows only full second-order models to be investigated over hyperspherical design regions, using an optimization strategy which combines a Nelder-Mead simplex search with a grid search. In contrast, **VdgRsm** employs random sampling to explore design regions. **VdgRsm** also includes FDS plots. However, **VdgRsm** allows only for full second-order models, and the user has limited control over the graphical output.

In this paper we present and discuss the new R package **vdg** (Schoonees 2016), of which version 1.2.0 is available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=vdg`. The aims of this package are threefold. First, we aim to offer an implementation of VDGs and FDS plots which harness the modern graphical capabilities of R by utilizing the popular **ggplot2** plotting system (Wickham 2009; Wickham and Chang 2016). Furthermore, **vdg** offers generalizations of VDGs by borrowing ideas from QPs (Khuri *et al.* 1996) and increased flexibility by harnessing the well-known R modelling infrastructure. The latter use of formulae and model matrices greatly increase the scope of models that can be explored. Finally, **vdg** uses random sampling as a design philosophy for exploring design regions, in contrast to optimization methods employed in traditional VDG software. This allows novel enhancements of VDGs such as allowing for flexible design regions, while also alleviating convergence problems and lack of flexibility often characteristic of traditional software tools in this area.

In the subsequent section the theory of these graphical procedures is briefly reviewed, and in particular some extensions are offered. This is followed by a succinct overview of sampling algorithms available in our package. Finally, the **vdg** package is introduced together with a discussion of its main features, and several examples are considered.

2. Graphical design assessment techniques and extensions

Let ξ_n denote an n-run exact experimental design, and suppose m design variables are being considered. The design runs are gathered in the $n \times m$ matrix $\mathbf{X}$. Consider a regression model linear in p parameters that requires the expansion of each design run $\mathbf{x}_i : m \times 1$ into a model vector $\mathbf{f}(\mathbf{x}_i) : p \times 1$, and define $\mathbf{F} : n \times p$ as the design matrix with these vectors as rows. For example, when $m = 2$ and $p = 4$, consider the model

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2.$$

This model requires the expansion of $\mathbf{x} = (x_1, x_2)^\top$ into $\mathbf{f}(\mathbf{x}) = (1, x_1, x_2, x_1 x_2)^\top$ where $\mathbf{x}^\top$ denotes the transpose of the column vector $\mathbf{x}$. The Fisher information matrix $\mathbf{M}(\xi_n)$ is given by

$$\mathbf{M}(\xi_n) = n^{-1} \mathbf{F}^\top \mathbf{F}. \tag{1}$$

Letting σ^2 denote the unknown error variance of the regression model, the variance of the estimated response at some $\mathbf{x}$ in the design space $\mathcal{X}$ under ordinary least squares estimation (OLS) is

$$\mathsf{VAR}(\hat{y}(\mathbf{x})) = \sigma^2 \, \mathbf{f}(\mathbf{x})^\top \left(\mathbf{F}^\top \mathbf{F} \right)^{-1} \mathbf{f}(\mathbf{x}). \tag{2}$$

Furthermore, define the scaled prediction variance (SPV) of the design ξ_n for a given model as

$$d(\mathbf{x}, \xi_n) = \sigma^{-2} \cdot n(\mathsf{VAR}(\hat{y}(\mathbf{x}))) = \mathbf{f}(\mathbf{x})^\top \mathbf{M}^{-1}(\xi_n) \mathbf{f}(\mathbf{x}). \tag{3}$$

Note that in Equations 1–3 only regression models that are linear in the parameters are considered. Models non-linear in the parameters have the property that the Fisher information matrix, introduced below, is a function of the unknown parameters. In such models the design properties also depend on these unknown parameters, and designs for these models are consequently often called "locally optimal" in the optimal design context. In such cases a Bayesian approach to optimal design construction, as surveyed in Chaloner and Verdinelli (1995), is more natural. This falls outside the scope of **vdg**.

2.1. Variance dispersion graphs

The VDG of Giovannitti-Jensen and Myers (1989) displays a summary of the SPV of the design ξ_n over the design region $\mathcal{X}$. In the original formulation, the summary is achieved by concentric hyperspheres spanning $\mathcal{X}$ and usually centred at the origin. The VDG then summarizes the SPV distribution by plotting the estimated minimum, average and maximum SPV on the surface of each hypersphere against the radius of the hypersphere. In this way the high-dimensional SPV profile can always be summarized in a two-dimensional plot of the SPV values against the radius. An example is given in Figure 1, where $\mathcal{X}$ is spherical with maximum radius $\sqrt{3}$. Here the solid line represents the mean SPV, and the long and short dashed lines the maximum and minimum SPV respectively. The mean of the SPV distribution over the surface of a hypersphere is also known as the *spherical SPV*.

Due to the uncertainty regarding where in the design space a prediction will be required, a stable SPV profile is desirable. A stable SPV is associated with a relatively flat VDG and cases where the maximum and minimum SPV curves do not deviate markedly from the average SPV. This is not the case for the design in Figure 1, where the SPV increases greatly towards the boundary of $\mathcal{X}$. The minimum and maximum SPV allow the analyst to judge the

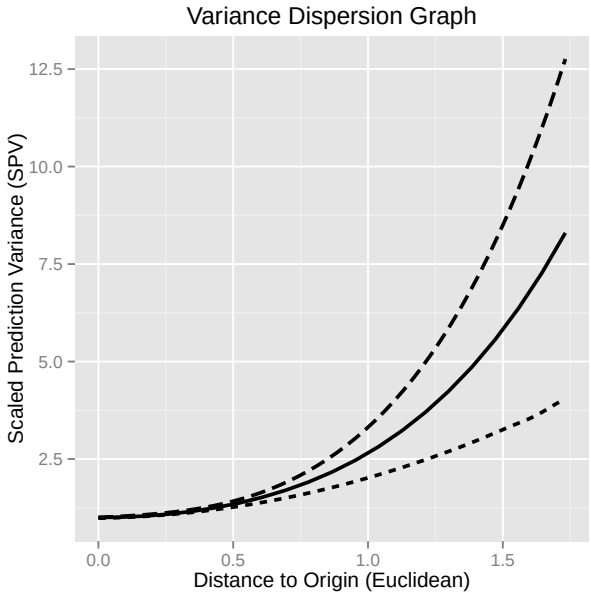


Figure 1: An example of a variance dispersion graph.

dispersion of the SPV about its average. Note that for rotatable designs the spherical SPV is constant for any given radius r, in which case the minimum and maximum SPV curves coincide with the average SPV curve. Quite clearly, the design in Figure 1 is not rotatable for the specified model. Rotatability can be advantageous as it is often the case that large deviations between minimum and maximum SPV occur near the perimeter of the experimental region. This is also the case in our example. Another obvious advantage of rotatability is that in such cases only the spherical SPV has to be computed. For non-rotatable designs the minimum and maximum SPV curves must be estimated by point-wise numerical optimization schemes, a task whose computational intensiveness can be aggravated by the presence of local optima.

The choice of concentric hyperspheres (i.e., radius) to summarize the SPV is related to the choice of Euclidean distance as a metric for summarizing the remoteness of the hypersphere from the origin (as in Figure 1). In practice any appropriate metric can be used. For hypercuboidal design regions using concentric hypercubes are indeed more natural. This relates to the choice of the L_∞-norm as metric (also known as supremum distance). We will see in Section 5 how **vdg** relies on the **proxy** package (Meyer and Buchta 2016) to incorporate a variety of metrics into the construction of VDGs.

Subsequently, the traditional VDG theory is discussed, assuming a hyperspherical design region.

VDG theory for hyperspherical design regions

Denote by $U_r = \{\mathbf{x} \in \mathcal{X} : \mathbf{x}^\top \mathbf{x} = r^2\}$ the *surface* of the hypersphere of radius r centred at the origin of the spherical design space $\mathcal{X}$. Giovannitti-Jensen and Myers (1989) define the

spherical SPV at r, i.e., the mean SPV over U_r, as:

$$\begin{aligned} V^r &= \frac{n\Theta}{\sigma^2} \int_{U_r} \mathsf{VAR}(\hat{y}(\mathbf{x}))\, d\mathbf{x} \\ &= n\Theta \mathrm{tr}\{(\mathbf{F}^\top \mathbf{F})^{-1} \int_{U_r} \mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x})^\top\, d\mathbf{x}\} \\ &= n\mathrm{tr}\{(\mathbf{F}^\top \mathbf{F})^{-1}\mathbf{S}\}. \end{aligned} \tag{4}$$

Here $\Theta^{-1} = \int_{U_r} d\mathbf{x}$ is the surface area of U_r and $\mathbf{S} : p \times p$ is the matrix of spherical region moments for U_r. Specifically, the elements of $\mathbf{S}$ have the form (Giovannitti-Jensen and Myers 1989):

$$\sigma(\boldsymbol{\delta}) = \Theta \int_{U_r} x_1^{\delta_1} x_2^{\delta_2} \ldots x_m^{\delta_m} d\mathbf{x}, \tag{5}$$

where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m)^\top$.

Due to the symmetry of the surface U_r, $\sigma(\boldsymbol{\delta})$ is zero whenever any of the $\{\delta_i\}$ are odd. Depending on the model, this property often leads to $\mathbf{S}$ being quite sparse. Often only a few elements of $\boldsymbol{\delta}$ are nonzero, and since the symmetry implies that it does not matter which elements these are, it is notationally more convenient to characterize the spherical region moments in Equation 5 in terms of the values of the nonzero elements. As an example, the two-variable first-order model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$ has a spherical region moment matrix of the form (Giovannitti-Jensen and Myers 1989):

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_2 \end{pmatrix}, \tag{6}$$

where

$$\sigma_2 = \frac{r^2}{m}. \tag{7}$$

For the model linear in the parameters, explicit equations for the spherical SPV function (as well as for the minimum and maximum SPV curves) exist (see Giovannitti-Jensen and Myers 1989). Note also that the second-order model in addition requires the following spherical region moments:

$$\sigma_4 = \frac{3r^4}{m(m+2)}, \tag{8}$$

$$\sigma_{22} = \frac{r^4}{m(m+2)}. \tag{9}$$

In this case the form of $\mathbf{S}$ is more involved (for further details, see Giovannitti-Jensen and Myers 1989).

Generalizing to more flexible models

However, to consider more general models, or models containing only specific terms, more general spherical region moments are required. For this purpose the VDG theory can be

extended as follows. The formula for a general spherical region moment with all elements of $\boldsymbol{\delta}$ even is:

$$\sigma(\boldsymbol{\delta}) = \frac{r^{\mathbf{1}^\top \boldsymbol{\delta}} \Gamma\left(\frac{m}{2}\right) \prod_{i=1}^{m} \Gamma\left(\frac{\delta_i+1}{2}\right)}{\pi^{\frac{m}{2}} \Gamma\left(\frac{\mathbf{1}^\top \boldsymbol{\delta}+m}{2}\right)}. \tag{10}$$

This formula can be derived from the evaluation of Equation 5 after using a transformation to polar coordinates (see for example Muirhead 1982, pp. 55–56) together with the properties of beta integrals. Considering the result in Equation 10, it is apparent that the spherical moments are functions of the radius r and the number of design variables m only. The higher-order moments dominate the average SPV at large radii since the factor $r^{\mathbf{1}^\top \boldsymbol{\delta}}$ in the formula is monotonically increasing. Equation 10 has been implemented in **vdg** to calculate the mean spherical SPV for any model formula (albeit subject to some restrictions – see `?meanspv` for details).

Advantages of the random sampling approach to VDGs

Note that although Equation 10 facilitates the analytical calculation of the average spherical SPV curve, the minimum and maximum SPV must still be found by pointwise numerical optimization techniques (except for first-order models – see Giovannitti-Jensen and Myers 1989). This can be a tedious exercise as typically optimization needs to be done over a grid of radii spanning the design region $\mathcal{X}$, and local optima may be problematic. The Fortran program of Vining (1993a,b) uses a combination of grid search and the Nelder-Mead simplex method, which can have erratic behavior.

Although R provides a wide variety of other optimization possibilities – see for example the CRAN Task View on Optimization and Mathematical Programming (Theussl and Borchers 2016) – using the random sampling approach to explore $\mathcal{X}$ with the computational power available holds many advantages. In fact we argue that there is no need to use the exact minimum or maximum to assess the spread of the SPV. Instead of using pointwise optimization, **vdg** can use quantile regression through many randomly sampled points to give the user an impression of how, e.g., the 5th and 95th percentiles evolve for different radii. For this we use the excellent **quantreg** package (Koenker 2016). At the same time the spread of the SPV at the randomly sampled points still gives an indication of the minimum or maximum SPV.

The form of the design region $\mathcal{X}$ is another important aspect to consider in the calculation of VDGs. If $\mathcal{X}$ is non-spherical, for example cuboidal, Equation 10 does not hold for larger radii r for which only a part of the hypersphere is contained in $\mathcal{X}$. In such a case only the part of the concentric hyperspheres contained in $\mathcal{X}$ should contribute to the minimum, average and maximum SPV curves, and truncation will be necessary. The ease with which the sampling approach can allow for these and other nonstandard design regions is an important advantage over the optimization approach.

Furthermore, it is important to note that the maximum and minimum SPV values plotted on the VDG only give an indication of the range of the spherical SPV. It is quite possible that two designs could have similar VDGs but quite different SPV distributions over the hyperspheres. In such a case, additional knowledge of the spherical SPV distribution is required, which is readily available when using the random sampling approach. A traditional approach of this flavour is the QP described in the next section.

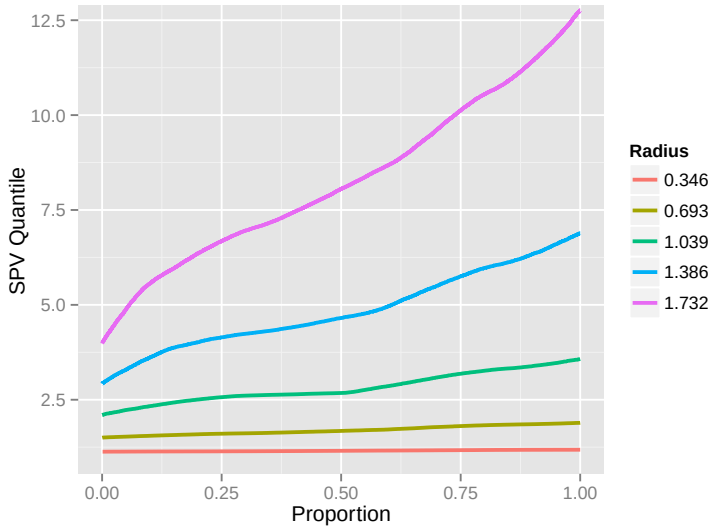


Figure 2: An example of a quantile plot, corresponding to the example in Figure 1.

2.2. Quantile plots

Related to the VDG is the QP of Khuri *et al.* (1996). QPs are based on the same principle as VDGs, but present more information about the distribution of the SPV over the hyperspheres. These plots use the quantiles of the SPV over the hyperspheres to display more information and can therefore be used to distinguish between designs with similar VDGs but with different SPV distributions.

In its original formulation, the basic idea of the QP is to graph the empirical cumulative distribution functions (or ECDFs) of the SPV over each hypersphere for a number of concentric hyperspheres with radii spanning the design region. These are estimated by randomly sampling a large number of points, say n^*, on the surface of each hypersphere. At each sampled point i the SPV d_i^r is calculated and the ECDF of $\{d_i^r\}$ is computed. The QP then consists of all the ECDFs. If the sampled points adequately cover the experimental region, the resulting graphs give detailed information about the distribution of the SPV over the experimental region. An example of such a QP is shown in Figure 2, where the quantiles of the SPV is plotted on the vertical axis. The lines in the plot represent the ECDFs for five concentric hyperspheres, which is a more refined way of representing the information in Figure 1.

An alternative version of the QP displays boxplots side by side to summarize the SPV distribution at different radii. This leads to a display similar to the corresponding VDG. In **vdg** we typically combine the information on display in a VDG and QP in a single graph (or a series of related graphs).

2.3. Fraction of design space plots

The FDS plot, introduced by Zahran *et al.* (2003), is based on the argument that accurate prediction over the entire $\mathcal{X}$ must take into account the fraction of the volume of $\mathcal{X}$ that is associated with the various values of the SPV. In contrast, the VDG and QP give equal weight to the SPV for all radii r, although the fraction of the volume of the design region represented by the hypersphere of each radius differs substantially. This means that a small

SPV at a radius close to the origin does not give as much assurance of good prediction as a similar SPV value at a larger radius.

The FDS plot is a way of correcting for this weighting problem which can make the use of the VDG and/or QP in isolation problematic. For example, it might occur that one design is far superior (in terms of the SPV) compared to another for a small to moderate r, but only somewhat inferior at large r. From the VDG it will be tempting to choose the first design since it is superior for most radii, but for higher dimensional design problems the design performance at the boundaries of the design region becomes exponentially more important. Therefore the second design may be preferable.

FDS theory

For a fixed value ν of the SPV $d(\mathbf{x}, \xi_n)$, the FDS criterion is defined as

$$FDS\left(\nu; \xi_n\right) = \Phi^{-1} \int_A d\mathbf{x}, \tag{11}$$

where $A = \{\mathbf{x} \in \mathcal{X} : d\left(\mathbf{x}, \xi_n\right) < \nu\}$ and Φ is the volume of $\mathcal{X}$. The criterion therefore expresses the volume of the set A containing all points in $\mathcal{X}$ with SPV lower than ν as a proportion of the total design space volume.

For each specified SPV value ν, the FDS plot graphs ν against FDS(ν). This in essence amounts to calculating the cumulative distribution function (CDF) for the SPV values and representing it graphically, albeit with a reversal of the usual horizontal and vertical axes (Goldfarb, Anderson-Cook, Borror, and Montgomery 2004). The latter reversal is motivated by the need to make the FDS plots directly comparable to VDGs.

Since calculating the FDS criterion analytically becomes cumbersome in higher dimensions, an ECDF is used in practice to estimate the CDF described above. This is done through Monte Carlo simulation. First, a large number of points, say n^*, throughout $\mathcal{X}$ is generated. For each of these points the SPV is computed and finally the ECDF is calculated. A design where the SPV values are relatively small and does not vary much over the design region is desirable.

Figure 3 is an example of an FDS plot for the running example in Figures 1 and 2. We see that for this design and model, the SPV is below 5 for 60% of the design region. The SPV exceeds 7.5 for roughly 12.5% of $\mathcal{X}$, with less that 5% of $\mathcal{X}$ having an SPV value in excess of 10.

Variants of FDS plots

Several variations of FDS plots exist. One variation is to use the so-called unscaled prediction variance (or UPV) instead of the SPV of Equation 3. The UPV $d^*(\mathbf{x}, \xi_n)$ is defined as

$$d^*(\mathbf{x}, \xi_n) = n \cdot d(\mathbf{x}, \xi_n) = \sigma^{-2}\, \mathsf{VAR}(\hat{y}(\mathbf{x})). \tag{12}$$

The UPV is appropriate when the size and the related cost of two or more designs are not considered to be important. This can occur when the cost of individual runs are insignificant.

Another version of FDS plots is the scaled FDS plot which focuses on the stability of the prediction variance over the design region $\mathcal{X}$ (Zahran 2002). Here the normal FDS plot is

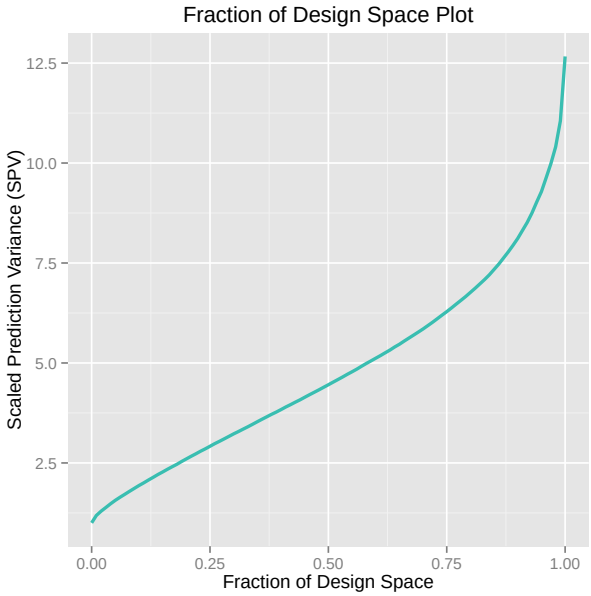


Figure 3: An example of an FDS plot, corresponding to Figures 1 and 2.

adjusted by using the scaled SPV, which can be expressed as

$$\frac{d(\mathbf{x}, \xi_n)}{\min_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x}, \xi_n)}. \tag{13}$$

A design with a steeper scaled SPV curve as compared to another design has a SPV which is less stable over the design region. This version of the FDS plot also allows the analyst to read off the ratio of the maximum to the minimum SPV.

One further variant is the variance ratio FDS plot (or VRFDS; Rodriguez, Jones, Borror, and Montgomery 2010). This plot is especially useful when a number of candidate designs are being compared to a reference design. To construct such a plot, the SPV is calculated for a number of different designs at the same randomly simulated points over the design region $\mathcal{X}$. The VRFDS plot is then constructed by replacing the SPV with the natural logarithm of the ratio of the SPV of each design to the SPV of the reference design. Supposing that two designs, $\xi^1_{n_1}$ and $\xi^2_{n_2}$ are of interest, the VRFDS plot is constructed by calculating the log variance ratio

$$\log VR\left(\mathbf{x}^*, \xi^1_{n_1}; \xi^2_{n_2}\right) = \log \frac{d\left(\mathbf{x}^*, \xi^1_{n_1}\right)}{d\left(\mathbf{x}^*, \xi^2_{n_2}\right)} \tag{14}$$

for each simulated point $\mathbf{x}^*$.

In Equation 14, $\xi^2_{n_2}$ is the reference design and is represented by a constant log variance ratio of zero in the plot. If the log variance ratio for a design is negative, it has a lower SPV than the reference design and is therefore the preferred design. Similarly, when the log variance ratio is positive it has a larger SPV than the reference design and is therefore not preferred. A design which leads to better predictions over much of the experimental region will have a curve largely below the horizontal line representing the reference design. These VRFDS plots can be useful for eliminating designs which are not admissible in the sense that they perform worse compared to any other candidate design over most of the experimental region.

3. Simulation algorithms and guidelines

In this section the sampling algorithms used in the **vdg** package are introduced for exploring the design space $\mathcal{X}$. In addition, recommendations are provided regarding the number of samples required.

3.1. Simulation algorithms

It is important to note that random sampling of $\mathcal{X}$ has advantages compared to using a grid of points covering the design region. Borkowski (2003) shows that using a grid of points places too much emphasis on the boundary regions of $\mathcal{X}$, which leads to inaccurate SPV estimates since the SPV is likely to be large at the perimeter of $\mathcal{X}$. Although a similar accuracy can be achieved for a fine grid of points, the relative efficiency remains lower.

In the design literature generally cuboidal or spherical design regions are commonly employed. Therefore these two cases will receive special attention here. Note however that using random sampling to explore the design space implies that any type of design region can be considered as long as it is possible to sample uniformly over it.

Sampling in hypercubes

A trivial way to generate uniform samples in an m-dimensional hypercube is to generate each of the m coordinates independently as uniform random numbers. For $\mathcal{X} = [-a, a]^m$, the m-dimensional hypercube with sides of length $2a$, each coordinate is simply generated independently as uniform variates on $[-a, a]$.

Space-filling designs originate from the literature on computer experiments and provide an alternative to this method (Fang, Li, and Sudjianto 2006). Simulation studies for FDS graphs have shown that Latin hypercube sampling (LHS) can be more efficient for large design problems (Schoonees 2011). LHS was first introduced by McKay, Beckman, and Conover (1979). A Latin hypercube design is a design where each column of the $n \times m$ matrix $\mathbf{V}$ is a random permutation of the column levels $1, 2, \ldots, n$. This matrix is then transformed into the $n \times m$ design matrix $\mathbf{X}$, by adding a random uniform value to each column. LHS amounts to first dividing $\mathcal{X}$ into a grid of equally sized hypercuboidal blocks. Then exactly one block is selected from every dimension. Finally, a random uniform value is added to each selected block. An example of a LHS is given in Figure 4 and can be constructed as follows:

```
R> library("vdg")
R> set.seed(8745)
R> samp <- LHS(n = 10, m = 2, lim = c(-1, 1))
R> plot(samp, main = "", pty = "s", pch = 16, ylim = c(-1, 1),
+    asp = 1, xlab = expression(X[1]), ylab = expression(X[2]))
R> abline(h = seq(-1, 1, length.out = 10),
+    v = seq(-1, 1, length.out = 10), lty = 3, col = "grey")
```

Latin hypercube sampling thus provides an alternative to the grid procedure and the simple uniform random sampling outlined previously. It can be seen as an extension to stratified sampling as it ensures that every portion of the range of each design factor is represented. Fang *et al.* (2006) show that LHS produces samples with a smaller variance of the sample

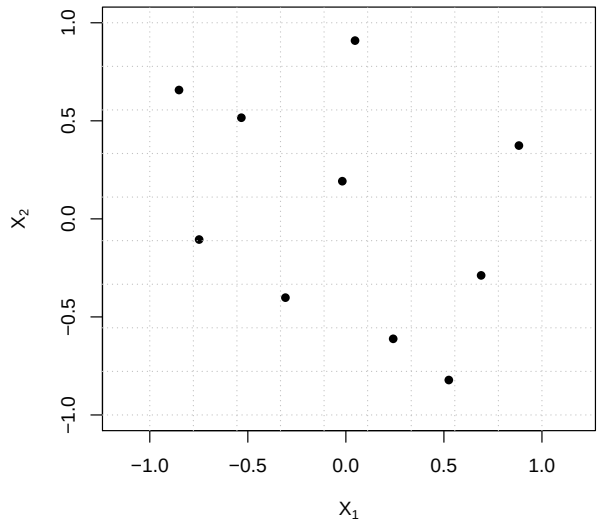


Figure 4: An example of an LHS of 10 points in a two-dimensional design space.

Algorithm 1 Latin hypercube sampling algorithm on $[-a, a]^m$ (cp. Fang *et al.* 2006)

- Calculate $\mathbf{b} = [-a + \frac{i}{n}], i = 1, 2, \ldots, n$.
- For $j = 1 \to m$:
  - Permute the elements of $\mathbf{b}$ to give $\mathbf{v}_j$.
  - Sample a vector of random numbers $\mathbf{u}_j$.
  - Calculate $\mathbf{x}_j$ as $\mathbf{v}_j - \frac{2a}{n} \cdot \mathbf{u}_j$.
- Form the matrix of samples as $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{pmatrix}$.

mean than simple random sampling. The interested reader should consult their book for a detailed discussion of the subject.

Algorithm 1 provides an overview of the method. Four different variants are implemented in **vdg** – see `?LHS` and the examples for more details. It is also possible to interface to other space-filling design implementations, such as those available in, for example, **lhs** (Carnell 2016) and **DiceDesign** (Dupuy, Helbert, and Franco 2015). An example is given in `?sampler`.

Sampling in hyperspheres

The obvious method for obtaining a uniform random sample inside a hypersphere is the rejection method – sample uniformly from the smallest hypercube containing the hypersphere and reject all samples falling outside the hypersphere. For higher dimensions and large samples this method is however inefficient. It is easy to show that the proportion of the volume of a hypercube, contained in the largest hypersphere, rapidly approaches zero as the number of dimensions m increases. It is therefore of interest to use alternative methods to generate

uniform random samples within hyperspheres.

This can conveniently be achieved in two independent steps. The first requires a uniform sample on the surface of the hypersphere with unit radius, after which each point is shrunken or extendend to the interior of the hypersphere by sampling radii from a particular distribution. The theory of spherical distributions, which includes the (multivariate) normal distribution, can be used to show that a point on the surface of the unit hypersphere can be found by sampling from a spherical distribution and rescaling the resulting vector to unit length (Schoonees 2011; Muirhead 1982). This is readily achieved by concatenating m univariate samples from, e.g., the normal distribution (`rnorm()` in R) and rescaling.

Furthermore, it can be shown that in order to ensure uniformity over the hypersphere, the CDF of the radius r on $[0, R]$ is given by:

$$F(r) = \frac{\Gamma(m/2+1)}{m\pi^{m/2}R^m} 2^{(m-1)(m-2)/2+1} r^m \prod_{i=1}^{m-2} B\left(\frac{m-i}{2}, \frac{m-i}{2}\right), \tag{15}$$

where $B(\cdot, \cdot)$ denotes the Beta function. The inverse CDF method is used in **vdg** to sample the required radii.

3.2. Simulation size

Ozol-Godfrey (2004) recommends using at least 10,000 randomly sampled points for an FDS graph of up to eight factors. In practice, it is not harmful in terms of computation time to use more. It should be clear from the plots produced by **vdg** whether or not a sufficient number of samples had been used.

4. The vdg package

The design of **vdg** revolves around the `spv()` function, which creates objects of S3 classes '`spv`', '`spvlist`', '`spvforlist`' and '`spvlistforlist`'. These classes differ with respect to the number of designs and model formulae passed to them. For example, an object of class '`spvlistforlist`' results from a call where both arguments `design` and `formula` are lists of designs and formulae repectively.

Objects of these classes contain the SPV (or unscaled prediction variance if `unscaled = TRUE`) evaluated for all designs and formulae at a set of n points sampled throughout $\mathcal{X}$. The SPV is calculated by a simple Fortran subroutine, and parallelization over multiple designs or formulae are facilitated by the base package **parallel**.

The sample can be passed explicitly by the user via the `sample` argument, but usually is constructed automatically by the `sampler()` function. The latter is a simple wrapper for the sampling algorithms built into the package (see Section 3), and automatically handles spherical and cuboidal design regions (via argument `type`). The user can request sampling on the surface of concentric hyperspheres or hypercubes by setting `at = TRUE` in a call to `spv()`. In such cases accurate FDS plots can, however, not be constructed and hence will not be produced. Rejection sampling for nonstandard design regions can be achieved by passing an appropriate function as the `keepfun` argument to `spv()`. In such cases sampling will continue until a sample of at least the requested size is obtained.

Besides simple `print()` methods, the power of **vdg** lies in its `plot()` methods, which produce a variety of graphs returned as a list of **ggplot2** objects. Use is made of facets to produce different panels for different designs and/or formulae. These graphical objects can subsequently be manipulated further, notably by using the theme functions in **ggplot2**. Which plots are produced depend on the input class as well as a variety of arguments (including `which = 1:5`). Only the most important can be mentioned here.

The `alpha` parameter determines the transparency of the plotted points, which helps to build a picture of the density of the prediction variance. It is often worth fine tuning this parameter, although a default is provided. The vector `tau` of values between zero and one determines which quantile regressions are included in the VDGs. A non-null value for the `radii` argument implies that the mean spherical prediction variance will be added to the VDGs according to Equation 10. It is advisable to read the help page of `meanspv()` when using this facility – there are limits to what types of `formula`s can be handled safely.

Finally, the optional `method` argument is passed to `proxy::dist()` and determines how the distance between a sampled point and the origin of $\mathcal{X}$ is determined. Several metrics are available, as described in `summary(pr_DB)` in **proxy**. If unset, the `type` argument will determine an appropriate value, namely Euclidean and supremum distance for spherical and cuboidal design regions respectively.

A starting point for exploring the package is `help("vdg-package")`. The call

```
R> vignette(topic = "vdg", package = "vdg")
```

will display a version of this paper which also serves as a package vignette.

5. Examples

5.1. Comparing four-factor hybrid designs

As a first example, we compare Roquemore's (Roquemore 1976) near-saturated 16-run hybrid designs D416B and D416C for a full second-order model in four factors. These designs are available in **Vdgraph** as

```
R> library("Vdgraph")
R> data("D416B", package = "Vdgraph")
R> data("D416C", package = "Vdgraph")
```

We can construct a VDG for these designs with **vdg** simply by

```
R> quad4 <- formula(~ (x1 + x2 + x3 + x4)^2 +  I(x1^2) + I(x2^2) +
+    I(x3^2) + I(x4^2))
R> set.seed(1234)
R> spv1 <- spv(n = 5000, design = list(D416B = D416B, D416C = D416C),
+    formula = quad4)
R> plot(spv1, which = "vdgboth")
```

Of course `quad4` could also have been specified more compactly as

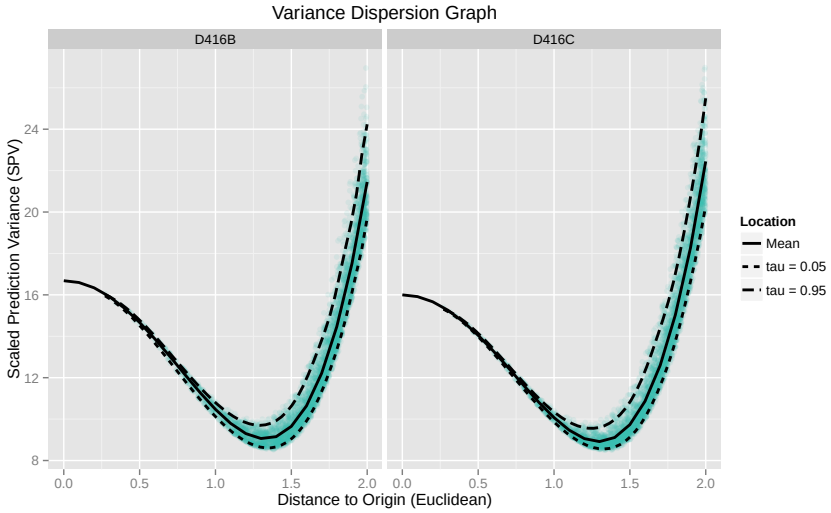


Figure 5: A VDG for Roquemore's hybrid designs D416B and D416C for a full quadratic model.

```
R> quad4 <- formula(~ .^2 +  I(x1^2) + I(x2^2) + I(x3^2) + I(x4^2))
```

Figure 5 shows the resulting VDG for these two designs, where the 5th and 95th percentiles are shown as quantile regression fits, together with the mean spherical SPV. The individual sampled points are overplotted, which gives an indication of the distribution of the SPV over the design region. It is evident that the SPV profiles for the hyperspherical design region $\mathcal{X}$ are very similar for both designs. D416B has a slightly higher SPV in proximity of the origin, but performs better near the perimeter of the design space. Hence it can be expected that D416B is preferable to D416C for this model and design space. This is confirmed by the FDS plots constructed by

```
R> plot(spv1, which = "fds")
R> plot(spv1, which = "fds", VRFDS = TRUE, np = 100)
```

as shown in Figure 6. The right panel shows that D416B is slightly superior over the majority of the design region.

Additional options to the `which` argument are outlined in `?plot.spv`. Multiple plots can be produced in a single call by passing a vector to `which`. Note that the plots are by default returned as a list of **ggplot2** graphical objects (see, e.g., Murrell 2011). The plots are rendered when they are `print()`ed, which implies that on some graphics devices only the last plot will be visible. This can be dealt with by either storing the resulting list of graphical objects and rendering them one by one, or by specifying `arrange = TRUE` in the `plot()` call. The latter will arrange all plots in a single device by using `grid.arrange()` from the **gridExtra** package (Auguie 2016). Yet another option is to set `par(ask = TRUE)` before creating the plots, which will prompt the user before advancing to the next plot.

An advantage of storing the plots as graphical objects is that we can make post-hoc changes before rendering the plot. For example, we might not like the default **ggplot2** theme used for Figure 5. We can easily change the theme to something more traditional with

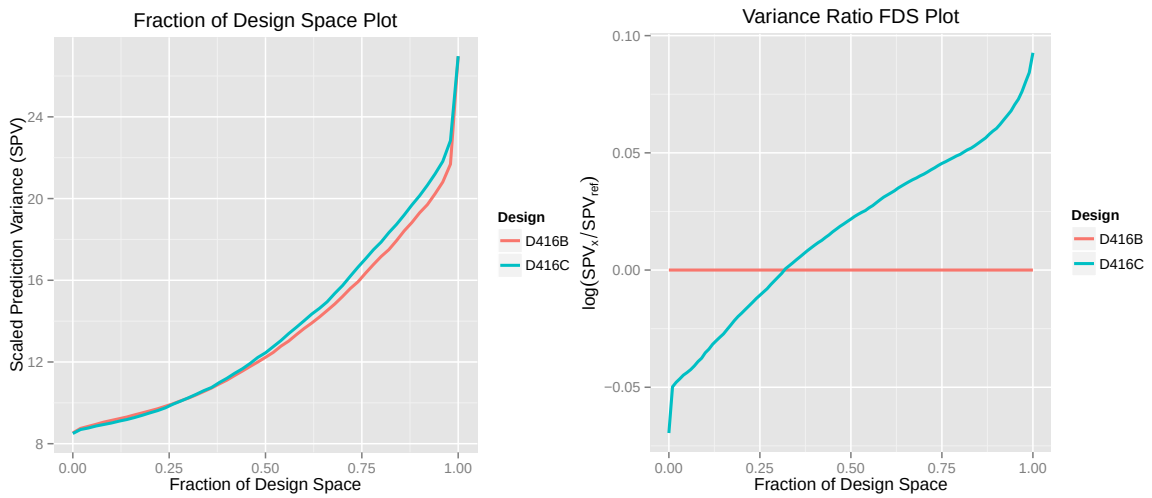


Figure 6: A standard and variance ratio FDS plot for Roquemore's hybrid designs D416B and D416C for a full quadratic model.

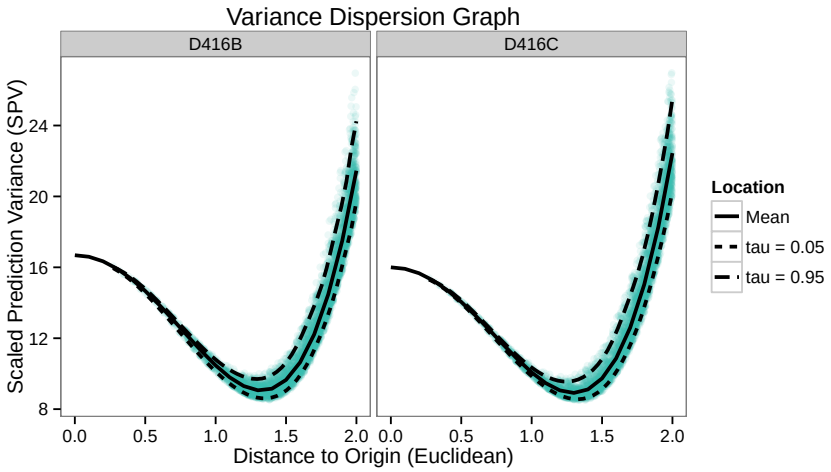


Figure 7: A second version of Figure 5.

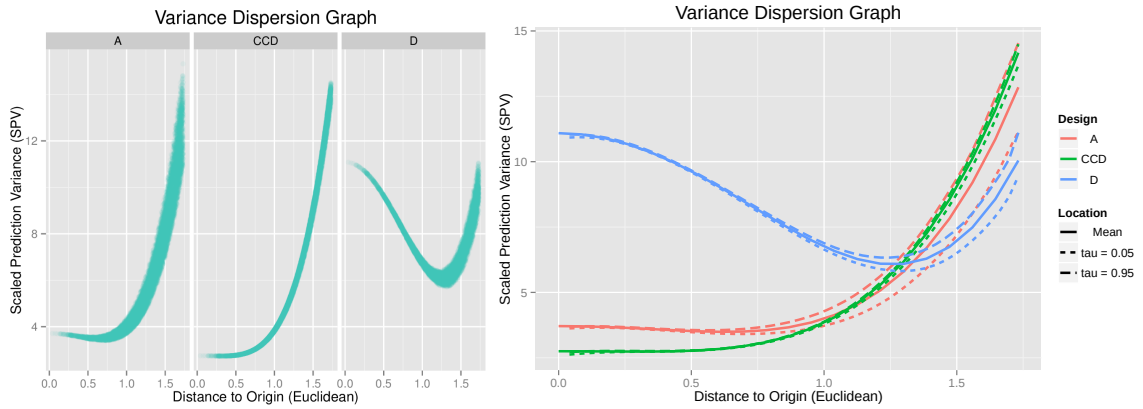
```
R> p <- plot(spv1, which = "vdgboth")
R> p$vdgboth + theme_bw() + theme(panel.grid = element_blank())
```

The result is shown in Figure 7. Much more refined approaches are also possible; see, for example, `?ggplot2::theme` and Wickham (2009).

5.2. Central composite, D- and A-optimal designs

In this example we consider optimal design alternatives to central composite designs (CCDs) for three design factors. A hyperspherical design region is assumed, and the axial distance for the CCD is assumed to be $\alpha = \sqrt{3}$. This spherical CCD is based on a full factorial design, augmented with four center runs and the usual six axial runs. The CCD hence contains 22 runs, and we can construct it using **rsm** (Lenth 2009) as

```
R> library("rsm")
```



(a) (b)

Figure 8: A FDS plot and VDG for the CCD, D- and A-optimal designs, for a full quadratic model in three factors on a spherical design region.

```
R> ccd3 <- as.data.frame(ccd(basis = 3, n0 = 4, alpha = "spherical",
+     oneblock = TRUE))[, 3:5]
```

We also construct a D- and A-optimal design with `optFederov()` from **AlgDesign** (Wheeler 2014). For this purpose a candidate list of 10 000 randomly sampled points is constructed within the sphere of radius α, with `runif_sphere()` in **vdg**:

```
R> set.seed(8619)
R> cand <- runif_sphere(n = 10000, m = 3)
R> colnames(cand) <- colnames(ccd3)
```

The algorithm then attempts to select the 22 runs from these candidates which optimize the requested criterion. D-optimality seeks to maximize the determinant of the information matrix in Equation 1, which implies minimizing the volume of the confidence ellipsoid of the regression parameters. A-optimality seeks to minimize the trace of the inverse information matrix, which minimizes the average variance of the regression parameters (for more details see, for example, Myers *et al.* 2009). The D- and A-optimal designs can be found by executing

```
R> quad3 <- formula(~ (x1 + x2 + x3)^2 + I(x1^2) + I(x2^2) + I(x3^2))
R> library("AlgDesign")
R> set.seed(3476)
R> desD <- optFederov(quad3, data = cand, nTrials = 22, criterion = "D")
R> desA <- optFederov(quad3, data = cand, nTrials = 22, criterion = "A")
```

All the information needed for the plots is obtained by calling `spv()` as

```
R> spv2 <- spv(n = 10000, formula = quad3,
+     design = list(CCD = ccd3, D = desD$design, A = desA$design))
R> plot(spv2, which = 2:3)
```

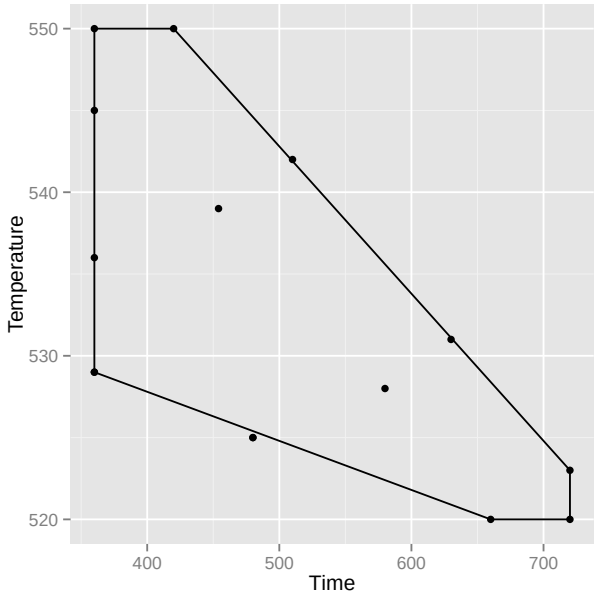


Figure 9: The design region and D-optimal design of Goos and Jones (2011). Some runs are replicated.

The VDG and FDS plots are shown in Figure 8. From the VDG it is apparent that the CCD and A-optimal designs have similar SPV profiles over $\mathcal{X}$, with SPV being very low near the origin but becoming increasingly worse towards the perimeter. The D-optimal design does worse in the interior but at the same time has better SPV near the perimeter of $\mathcal{X}$.

The FDS plot shows that the D-optimal design has the most stable SPV. Although the CCD and A-optimal designs achieve lower SPV values near the origin, this comes at the cost of a significantly higher SPV near the perimeter of $\mathcal{X}$. The FDS plot places more emphasis on the perimeter since this is where the majority of the volume of $\mathcal{X}$ is located. Hence the D-optimal design provides an alternative to the CCD which features better prediction over the majority of the design region.

5.3. Cubic model with restricted design region

Chapter 5 of Goos and Jones (2011) describes an experimental problem in two variables, namely time (seconds) and temperature (degrees Kelvin), for a chemical reaction in an industrial setting. The aim was to refine the current operating conditions to optimize the yield of the chemical process. However, based on prior knowledge the experimental region $\mathcal{X}$ has a restricted shape, as shown in Figure 9. Outside this region the combination of design factors was deemed not worth exploring by researchers with intimate knowledge of the process.

Furthermore, a full cubic model was decided on since a quadratic model was deemed inadequate to capture the nonlinearity of the response process. A 15-run design was feasible – the D-optimal design employed by Goos and Jones (2011, Table 5.4) and shown in Figure 9 is included in **vdg** as `GJ54` (not all design points lie strictly within $\mathcal{X}$). In order to construct a VDG and FDS plot for this design, we need to be able to sample from $\mathcal{X}$. Translating to standardized coordinates on $[-1, 1]$ in both time and termperature (using, e.g., `stdrange()`

in **vdg**), the equations of the linear restrictions on $\mathcal{X}$ are:

$$y = -1.08x + 0.28,$$
$$y = -0.36x - 0.76.$$

We can use this to construct a function `keepfun()` which takes a data matrix `x` and returns a logical vector indicating which of the samples in the rows of `x` lie within $\mathcal{X}$:

```
R> keepfun <- function(x) apply(x >= -1 & x <= 1, 1, all) &
+    (x[, 2] <= -1.08 * x[, 1] + 0.28) & (x[, 2] >= -0.36 * x[, 1] - 0.76)
```

We compare the SPV for the full cubic model, for which the design was constructed, to this model without the cubic term for time and the interaction between temperature and the quadratic term in time. This latter model was the final model after removing the two insignificant effects – see Goos and Jones (2011). The formula for this model is obtained as

```
R> cube2 <- formula(~ (Time + Temperature)^2 + I(Time^2) +
+    I(Temperature^2) + I(Time^3) + I(Temperature^3) +
+    Time:I(Temperature^2) + I(Time^2):Temperature)
R> GJmod <- update(cube2, ~ . - I(Time^3) - I(Time^2):Temperature)
```

To sample uniformly over $\mathcal{X}$, we can use `keepfun()` to construct an '`spv`' object for the standardized design. The algorithm will automatically keep sampling until at least the specified number of samples have been obtained. We now pass a list of model formulae to `spv()`, and use Latin hypercube sampling for illustration:

```
R> spv3 <- spv(n = 10000, design = stdrange(GJ54), type = "lhs",
+    formula = list(Cubic = cube2, GoosJones = GJmod), keepfun = keepfun)

Retained samples: 5016 -- Adding some more...
Retained samples: 9976 -- Adding some more...
Final sample of size 10490

R> plot(spv3, which = 1, points.size = 2)
```

Of course the SPV surface for the second model is much favoured to the full cubic model, but the unimportance of the dropped terms is difficult if not impossible to establish before constructing a design. Note in Figure 10 that supremum distance is automatically selected to summarize the SPV since the argument `type = "cuboidal"` was specified. This can be changed to, e.g., Manhattan distance by passing `method = "Manhattan"` to the call to `plot()`. Users are encouraged to experiment with the option `hexbin = TRUE` which uses **ggplot2**'s hexagonal binning feature as an alternative to overplotting.

Since this example has only two variables, the SPV surface can also be inspected by contour plots, or in three dimensions by, e.g., **rgl** (plot not shown; Adler, Murdoch, and others 2016):

```
R> library("rgl")
R> with(spv3$Cubic, plot3d(x = sample[, "Time"], y = sample[, "Temperature"],
+    z = spv))
```

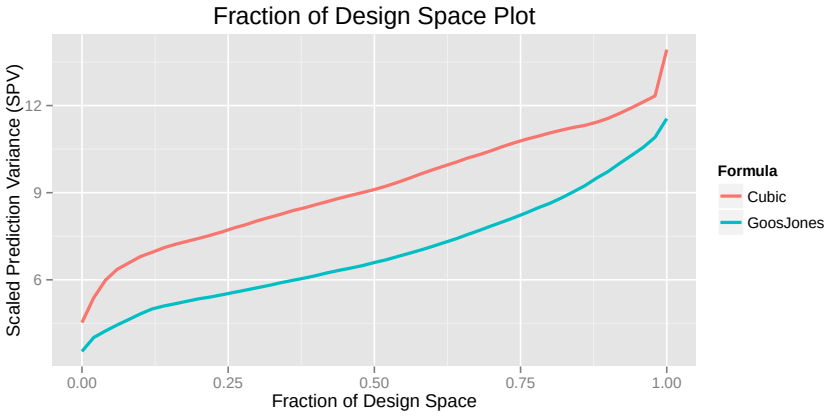


Figure 10: VDG for the D-optimal design of Goos and Jones (2011), for the two models.

6. Conclusions

The **vdg** package provides a modern and flexible R interface to important graphical procedures in experimental design, producing VDGs, FDS and related plots using random sampling. Multiple designs and/or multiple model formulae are seamlessly allowed for, while using R's `formula` interface allows for flexibility in model specification. Plots are produced with the **ggplot2** package, which allows for post hoc manipulation of graphical elements.

The flexibility of **vdg** empowers users to investigate irregular design regions; to use different sampling schemes; to consider complicated model specifications; to construct different variants of the standard plots, such as the VRFDS plot, as well as to enhance plots with novelties like quantile regressions. In addition to its capabilities illustrated in the examples given above, the plot method of **vdg** includes features like specifying specific distance measures from the **proxy** package via the `method` argument and using hexagonal binning via `hexbin = TRUE` instead of overplotting. Thus, **vdg** not only plays a complimentary role to the collection of existing R packages available to the practitioner in the field of experimental designs, but adds worthwhile capabilities for use in this field.

References

Adler D, Murdoch D, others (2016). ***rgl**: 3D Visualization Device System (OpenGL)*. R package version 0.96.0, URL https://CRAN.R-project.org/package=rgl.

Atkinson A, Donev A, Tobias R (2007). *Optimum Experimental Designs, with SAS*. Oxford University Press.

Auguie B (2016). ***gridExtra**: Miscellaneous Functions for **grid** Graphics*. R package version 2.2.1, URL https://CRAN.R-project.org/package=gridExtra.

Borkowski JJ (2003). "A Comparison of Prediction Variance Criteria for Response Surface Designs." *Journal of Quality Technology*, **35**(1), 70–77.

Carnell R (2016). ***lhs**: Latin Hypercube Samples*. R package version 0.14, URL https://CRAN.R-project.org/package=lhs.

Chaloner K, Verdinelli I (1995). "Bayesian Experimental Design: A Review." *Statistical Science*, **10**(3), 273–304. doi:10.1214/ss/1177009939.

Dupuy D, Helbert C, Franco J (2015). "**DiceDesign** and **DiceEval**: Two R Packages for Design and Analysis of Computer Experiments." *Journal of Statistical Software*, **65**(11), 1–38. doi:10.18637/jss.v065.i11.

Fang K, Li R, Sudjianto A (2006). *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC. doi:10.1201/9781420034899.

Giovannitti-Jensen A, Myers R (1989). "Graphical Assessment of the Prediction Capability of Response Surface Designs." *Technometrics*, **31**(2), 159–171. doi:10.1080/00401706.1989.10488510.

Goldfarb H, Anderson-Cook C, Borror C, Montgomery D (2004). "Fraction of Design Space Plots for Assessing Mixture and Mixture-Process Designs." *Journal of Quality Technology*, **36**(2), 169–179.

Goos P, Jones B (2011). *Optimal Design of Experiments: A Case Study Approach*. John Wiley & Sons. doi:10.1002/9781119974017.

Khuri A, Kim H, Um Y (1996). "Quantile Plots of the Prediction Variance for Response Surface Designs." *Computational Statistics & Data Analysis*, **22**(4), 395–407. doi:10.1016/0167-9473(95)00058-5.

Koenker R (2016). ***quantreg**: Quantile Regression*. R package version 5.29, URL https://CRAN.R-project.org/package=quantreg.

Lawson J (2014). *The **Vdgraph** Package*. R package version 2.2-2, URL https://CRAN.R-project.org/package=Vdgraph.

Lenth R (2009). "Response-Surface Methods in R, Using **rsm**." *Journal of Statistical Software*, **32**(7), 1–17. doi:10.18637/jss.v032.i07.

McKay M, Beckman R, Conover W (1979). "Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics*, **21**(1), 239–245. doi:10.1080/00401706.1979.10489755.

Meyer D, Buchta C (2016). ***proxy**: Distance and Similarity Measures*. R package version 0.4-16, URL https://CRAN.R-project.org/package=proxy.

Muirhead R (1982). *Aspects of Multivariate Statistical Theory*. John Wiley & Sons. doi:10.1002/9780470316559.

Murrell P (2011). *R Graphics*. 2nd edition. Chapman & Hall/CRC. doi:10.1201/9781420035025.

Myers R, Montgomery D, Anderson-Cook CM (2009). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons.

Ozol-Godfrey A (2004). *Understanding Scaled Prediction Variance Using Graphical Methods for Model Robustness, Measurement Error and Generalized Linear Models for Response Surface Designs*. Ph.D. thesis, Virginia Polytechnic Institute and State University.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rodriguez M, Jones B, Borror C, Montgomery D (2010). "Generating and Assessing Exact G-Optimal Designs." *Journal of Quality Technology*, **42**(1), 3–20.

Roquemore K (1976). "Hybrid Designs for Quadratic Response Surfaces." *Technometrics*, **18**(4), 419–423. doi:10.1080/00401706.1976.10489473.

Schoonees P (2011). *Sequential Experimental Design Strategies for Second-Order Models*. Master's thesis, University of Stellenbosch, South Africa.

Schoonees P (2016). ***vdg**: Variance Dispersion Graphs and Fraction of Design Space Plots*. R package version 1.2.0, URL https://CRAN.R-project.org/package=vdg.

Srisuradetchai P, Borkowski JJ (2015). ***VdgRsm**: Variance Dispersion and Fraction of Design Space Plots*. R package version 1.5, URL https://CRAN.R-project.org/package=VdgRsm.

Theussl S, Borchers HW (2016). "CRAN Task View: Optimization and Mathematical Programming." Version 2016-06-06, URL https://CRAN.R-project.org/view=Optimization.

Vining G (1993a). "A Computer Program for Generating Variance Dispersion Graphs." *Journal of Quality Technology*, **25**(1), 45–58.

Vining G (1993b). "Corrigenda to "A Computer Program for Generating Variance Dispersion Graph"." *Journal of Quality Technology*, **25**(1), 333–335.

Wheeler B (2014). ***AlgDesign**: Algorithmic Experimental Design*. R package version 1.1-7.3, URL https://CRAN.R-project.org/package=AlgDesign.

Wickham H (2009). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York.

Wickham H, Chang W (2016). ***ggplot2**: An Implementation of the Grammar of Graphics*. R package version 2.1.0, URL https://CRAN.R-project.org/package=ggplot2.

Zahran A (2002). *On the Efficiency of Designs for Linear Models in Non-Regular Regions and the Use of Standard Designs for Generalized Linear Models*. Ph.D. thesis, Virginia Polytechnic Institute and State University.

Zahran A, Anderson-Cook C, Myers R (2003). "Fraction of Design Space to Assess Prediction Capability of Response Surface Designs." *Journal of Quality Technology*, **35**(4), 377–386.

Affiliation:

Pieter C. Schoonees
Department of Marketing Management
Rotterdam School of Management, Erasmus University
3062PA Rotterdam, The Netherlands
E-mail: schoonees@rsm.nl
URL: http://www.rsm.nl/

Niël J. le Roux
Department of Statistics and Actuarial Science
University of Stellenbosch
Stellenbosch, South Africa
E-mail: njlr@sun.ac.za
URL: http://academic.sun.ac.za/statistics

Roelof L.J. Coetzer
Group Technology
Sasol South Africa
Sasolburg, South Africa
E-mail: roelof.coetzer@sasol.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2016, Volume 74, Issue 3 *Submitted:* 2014-07-04
doi:10.18637/jss.v074.i03 *Accepted:* 2015-08-24