APTS Design of Experiments

September 2019

Dave Woods (D.Woods@southampton.ac.uk ; http://www.southampton.ac.uk/~davew)
Statistical Sciences Research Institute, University of Southampton

Preliminaries

A few example designs and data sets for this module are available in the R package apts.doe, which can be installed from GitHub

```
library(devtools)
install_github("statsdavew/apts.doe", quiet = T)
library(apts.doe)
```

References will be provided throughout but some good general purpose texts are

- Atkinson, Donev and Tobias (2007). Optimum Experimental Design, with SAS. OUP
- Wu and Hamada (2009). Experiments: Planning, Analysis, and Parameter Design Optimization (2nd ed.). Wiley.
- Morris (2011). Design of Experiments: An Introduction based on Linear Models. Chapman and Hall/CRC Press.
- Santner, Williams and Notz (2019). The Design and Analysis of Computer Experiments (2nd ed.). Springer.

These notes and other resources can be found at https://statsdavew.github.io/apts.doe/

Motivation and background

Modes of data collection

- Observational studies
- Sample surveys
- **Designed experiments**

Experiments

Definition: An experiment is a procedure whereby controllable factors, or features, of a system or process are **deliberately** varied in order to understand the impact of these changes on one or more measurable responses.

- "prehistory": Bacon, Lind, Peirce, … (establishing the scientific method)
- agriculture (1920s)
- clinical trials (1940s)
- industry (1950s)
- in-silico (1980s)

Role of experimentation

Why do we experiment?

- key to the scientific method
  (hypothesis – **experiment** – observe – infer – conclude)
- potential to establish **causality** …
- … and to understand/improve complex systems depending on many factors
- comparison of treatments, factor screening, prediction, optimisation, …

Design of experiments: a statistical approach to the arrangement of the operational details of the experiment (eg sample size, specific experimental conditions investigated, …) so that the quality of the answers to be derived from the data is as high as possible.

Motivating examples

1. Multi-factor experiment in pharmaceutical development.

Key to developing new medicines is the identification of optimal **and robust** process conditions (e.g. settings of temperature, pressure etc.) at which the active pharmaceutical ingredient should be synthesized.

[Somewhat confusinging, the FDA refer to this as identification of a "design space".]

An important step in is this methodology is a robustness experiment to assess the sensitivity of identified conditions to changes in all (or at least very many) controllable factors.

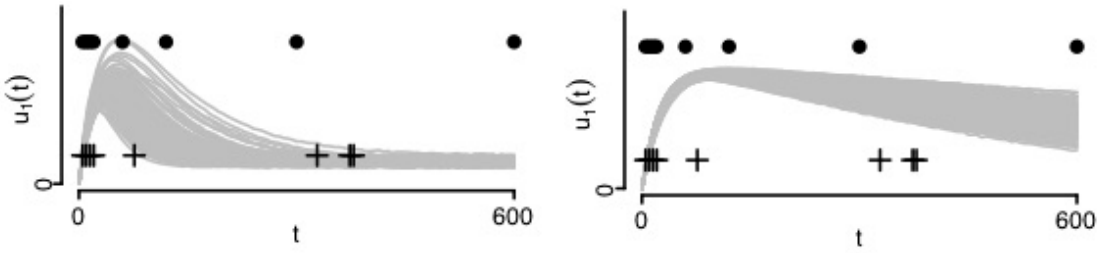
While developing a new melanoma drug, GlaxoSmithKline performed an experiment to investigate sensitivity to 20 factors. Their experimental budget allowed only 10 individual experiments (runs) to be performed.

Motivating examples

2. Optimal design to calibrate a physical model.

Physical (mechanistic, mathematical, …) models are used in many scientific fields. Typically, they are derived from fundamental understanding of the physics, chemistry, biology …

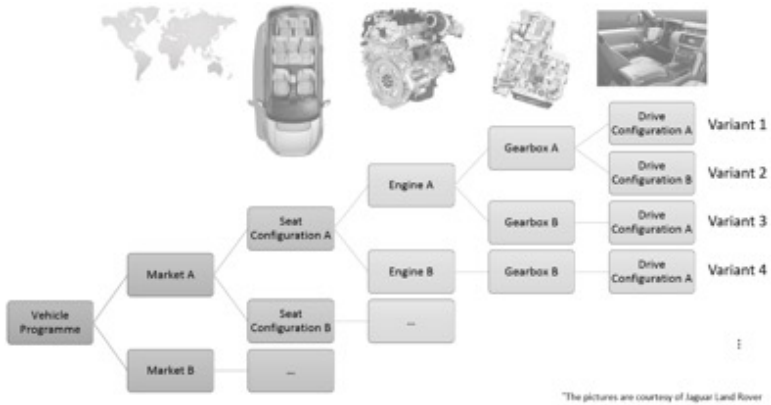
Most commonly, these models are solutions to differential equations. The models usually contain unknown parameters that should be estimated from experimental data.



Biologists at Southampton were studying the transfer of amino acids between mother and baby through the placenta. They could control the times at which observations were taken and the initial concentrations of amino acids (see Overstall, Woods, and Parker 2019).

Motivating examples

3. Computer experiments to optimise ride performance in luxury cars



Suspension settings can be used to improve the ride performance in cars. Optimising settings across many different car models would take many hundreds of hours of testing, so computer simulations are used.

Jaguar-Land Rover wanted to find suspension settings **robust** across different car models using a computer experiment (KTN workshop).

Simple motivating example

Consider an experiment to compare two treatments (eg drugs, diets, fertilisers).

We have n subjects (eg people, mice, plots of land), each of which can be assigned to one of the two treatments.

A response (eg protein measurement, weight, yield) is then measured from each subject.

Question: How should the two treatments be assigned to the subjects to gain the most precise inference about the difference in expected response from the two treatments.

Assume a linear model for the response

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i\,, \qquad i = 1, \ldots, n\,,$$

with $\varepsilon_i \sim N(0, \sigma^2)$ independently, β_0, β_1 unknown parameters and

$$x_i = \begin{cases} -1 & \text{if treatment 1 is applied to subject } i\,, \\ +1 & \text{if treatment 2 is applied to subject } i \end{cases}$$

The difference in expected response between treatment 1 and 2 is

$$E(y_i \mid x_i = +1) - E(y_i \mid x_i = -1) = \beta_0 + \beta_1 - \beta_0 + \beta_1 = 2\beta_1$$

So we need the most precise possible estimator of β_1

Both β_0 and β_1 can be estimated using least squares (or equivalently maximum likelihood).

Writing

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}\,,$$

we obtain estimators

$$\hat{\boldsymbol{\beta}} = \left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}\boldsymbol{y}$$

with

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \left(X^{\mathrm{T}}X\right)^{-1}\sigma^2$$

In this simple example, we are interesting in estimating β_1 , and we have

$$\begin{aligned}\mathrm{Var}(\hat{\beta_1}) &= \frac{n\sigma^2}{n\sum x_i^2 - \left(\sum x_i\right)^2} \\ &= \frac{n\sigma^2}{n^2 - \left(\sum x_i\right)^2}\end{aligned}$$

Hence, we need to pick $x_1, \ldots, x_n$ to minimise $\left(\sum x_i\right)^2 = (n_1 - n_2)^2$

- denote as n_1 the number of subjects assigned to treatment 1, and n_2 the number assigned to treatment 2, with $n_1 + n_2 = n$
- it is obvious that $\sum x_i = 0$ if and only if $n_1 = n_2$

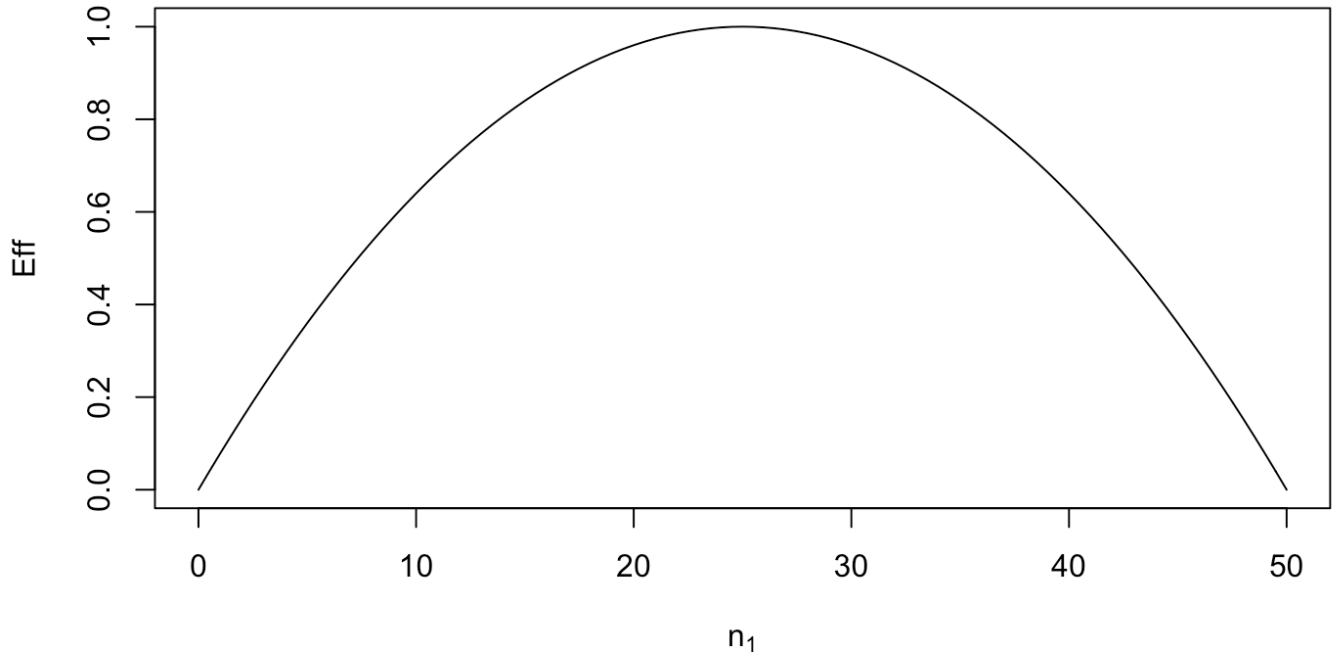
Assuming n is even, the "optimal design" has $n_1 = n_2 = n/2$

For n odd, let $n_1 = \frac{n+1}{2}$ and $n_2 = \frac{n-1}{2}$

We can assess a designs, labelled ξ, via its **efficiency** relative to the optimal design $\xi^\star$:

$$\text{Eff}(\xi) = \frac{\text{Var}(\hat{\beta_1} \mid \xi^\star)}{\text{Var}(\hat{\beta_1} \mid \xi)}$$

```r
n <- 50
eff <- function(n1) 1 - ((2 * n1 - n) / n)^2
curve(eff, from = 0, to = n, ylab = "Eff", xlab = expression(n[1]))
```



Definitions

- **Treatment** – entities of scientific interest to be studied in the experiment
  eg varieties of crop, doses of a drug, combinations of temperature and pressure
- **Unit** – smallest subdivision of the experimental material such that two units may receive different treatments
  eg plots of land, subjects in a clinical trial, samples of reagent
- **Run** – application of a treatment to a unit

Example

Fabrication of integrated circuits (see Wu and Hamada 2009)

- an initial step in fabricating integrated circuits is the growth of an epitaxial layer on polished silicon wafers via chemical deposition

Unit

- set of six wafers (mounted in a rotating cylinder)

Treatment

- combination of settings of the factors
  - A : rotation method (x_1)
  - B : nozzle position (x_2)
  - C : deposition temperature (x_3)
  - D : deposition time (x_4)

A unit-treatment statistical model

$$y_{ij} = \tau_i + \varepsilon_{ij}\,, \qquad i = 1, \ldots, t;\ j = 1, \ldots, n_i\,,$$

where

- y_{ij} : measured response from the jth unit to which treatment i has been applied
- τ_i : treatment effect (expected response from application of the ith treatment)
- ε_{ij} : random deviation from the expected response [typically $\sim N(0, \sigma^2)$]

The aims of the experiment are achieved by estimating comparisons between the treatment effects, $\tau_k - \tau_l$.

Experimental precision and accuracy are largely obtained through **control** and **comparison**.

Model assumptions

Three key model assumptions are:

- **additivity** (response = treatment effect + unit effect)
- **constancy** of treatment effects (treatment effect does not depend on the unit to which it is applied)
- **no interference** between units (the effect of a treatment applied to unit *j* does not depend on the treatment applied to any other unit)

See Dasgupta, Pillai, and Rubin (2015) for discussion of these assumptions for factorial experiments

Principles of experimentation

Stratification (blocking)

- account for systematic differences between batches of experimental units by arranging them in homogeneous sets (**blocks**)
  - if the same treatment was applied to all units, within-block variation in the response would be much less than between-block
  - compare treatments within the same block and hence eliminate block effects

Replication

- the application of each treatment to multiple experimental units
  - provides an estimate of experimental error against which to judge treatment differences
  - reduces the variance of the estimators of treatment differences

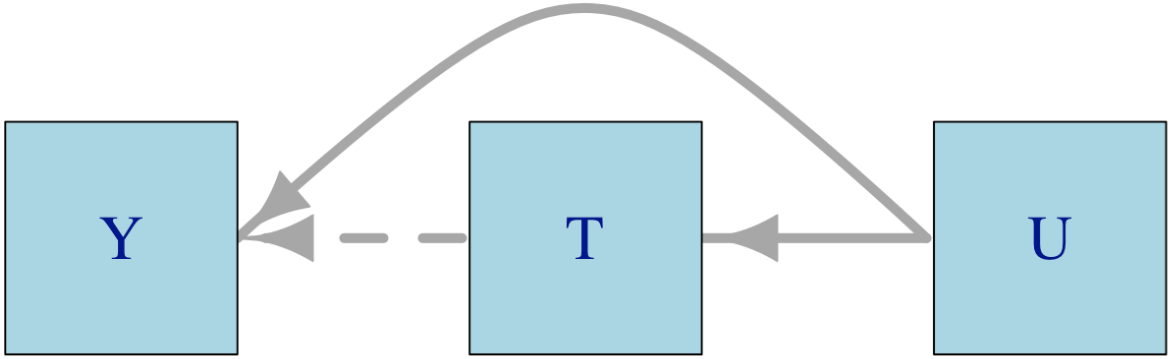
Randomisation

- we randomise features such as the allocation of units to treatments, the order in which treatments are applied, …
  - protects against lurking (uncontrolled) variables (model-robust) and subjectively in the allocation of treatments to units

Randomisation is perhaps the **key** principle in the design of experiments

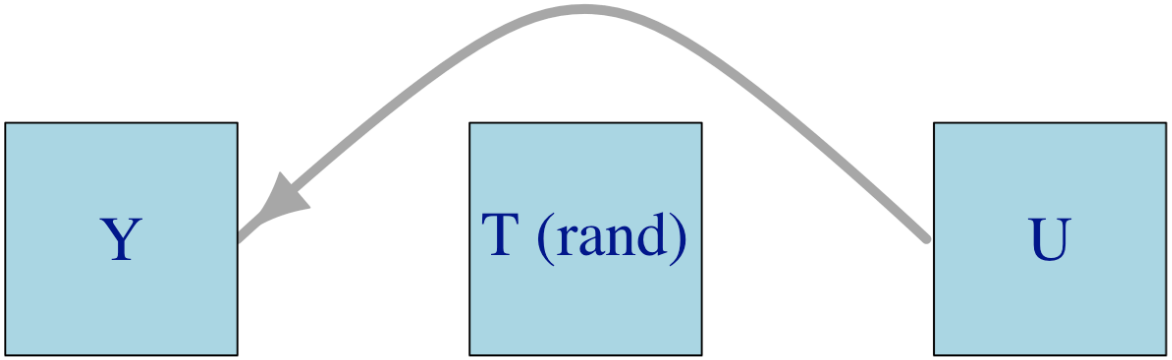
- it protects against model misspecification (bias), and hence allows **causality** to be established
  - a clear difference between treatments can only be an accident of the randomisation or a consequence of the treatments
- unbiased estimation of τ and σ^2, even if the errors are not normally distributed
- exact tests for differences between treatment effects are available (Basu 1980)

Without randomisation, unobserved confounders (U) can induce a dependency between the response (Y) and treatment (T)



cf Cox and Reid (2000), p.35

With randomisation, unobserved confounders (U) are independent of the treatment (T).
Marginalisation over U does not induce an edge between T and Y



cf Cox and Reid (2000), p.35

Factorial designs

Example revisited

Fabrication of integrated circuits (Wu and Hamada 2009, p155)

Treatment

- combination of settings of the factors
  - A : rotation method (x_1)
  - B : nozzle position (x_2)
  - C : deposition temperature (x_3)
  - D : deposition time (x_4)

Assume each factor has two-levels, coded -1 and +1

Treatments and a regression model

Each factor has two levels $x_k = \pm 1,\ k = 1, \ldots, 4$

A treatment is then defined as a combination of four values of $-1, +1$

- eg $x_1 = -1, x_2 = -1, x_3 = +1, x_4 = -1$
- specifies a setting of the process

Assume each treatment effect is determined by a regression model in the four factors, eg

$$\tau(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{4} \beta_i x_i + \sum_{j=1}^{4} \sum_{i>j}^{4} \beta_{ij} x_i x_j$$

(Two-level) Factorial design

```
with(cirfab, cirfab[order(x1, x2, x3, x4), ])
```

```
##    x1 x2 x3 x4     ybar
## 2  -1 -1 -1 -1 13.58983
## 1  -1 -1 -1  1 14.59000
## 4  -1 -1  1 -1 14.04983
## 3  -1 -1  1  1 14.24000
## 6  -1  1 -1 -1 13.94000
## 5  -1  1 -1  1 14.65000
## 8  -1  1  1 -1 14.14017
## 7  -1  1  1  1 14.40000
## 10  1 -1 -1 -1 13.72000
## 9   1 -1 -1  1 14.67000
## 12  1 -1  1 -1 13.90000
## 11  1 -1  1  1 13.84017
## 14  1  1 -1 -1 13.87983
## 13  1  1 -1  1 14.56000
## 16  1  1  1 -1 14.11017
## 15  1  1  1  1 14.30000
```

- treatments in standard order
- $\bar{y}$ - average response from the six wafers

Regression model and least squares

$$\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}\,, \qquad \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 I)\,, \qquad \hat{\boldsymbol{\beta}} = \left(X^{\mathrm{T}}X\right)^{-1} X^{\mathrm{T}}\boldsymbol{Y}$$

- model matrix X has columns corresponding to intercept, linear and cross-product terms
- information matrix $X^{\mathrm{T}}X = nI$
- regression coefficients are estimated by independent contrasts in the data

```
cirfab.lm <- lm(ybar ~ (.) ^ 2, data = cirfab)
coef(cirfab.lm)
```

```
##  (Intercept)           x1           x2           x3           x4        x1:x2
## 14.161250000 -0.038729167  0.086270833 -0.038708333  0.245020833  0.003708333
##        x1:x3        x1:x4        x2:x3        x2:x4        x3:x4
## -0.046229167 -0.025000000  0.028770833 -0.015041667 -0.172520833
```

Main effects and interactions

Main effect of x_k:

$$[\text{Avg. response when } x_k = 1] - [\text{Avg. response when } x_k = -1]$$

Interaction between x_j and x_k:

$$[\text{Avg. response when } x_j x_k = 1] - [\text{Avg. response when } x_j x_k = -1]$$

Higher-order interactions defined similarly

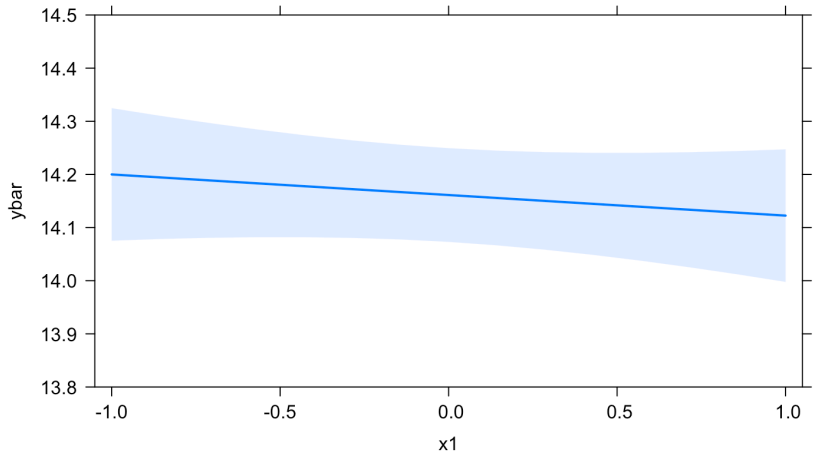
Assuming -1,+1 coding, there is a straightforward relationship between **factorial effects** and **regression coefficients**

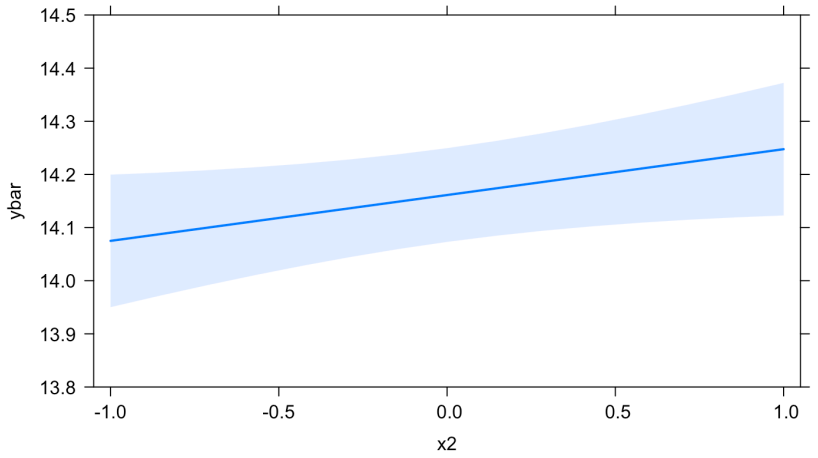
- main effect of x_k is equal to $2\beta_k$
- interaction between x_j and x_k is equal to $2\beta_{jk}$

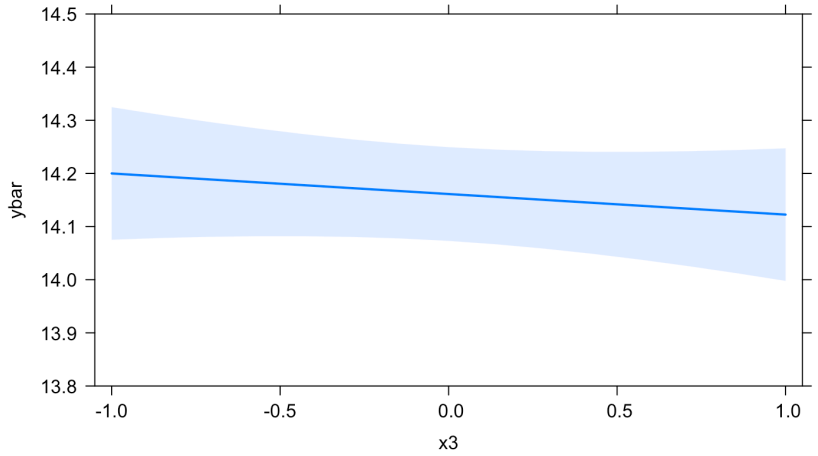
Using the `effects` package:

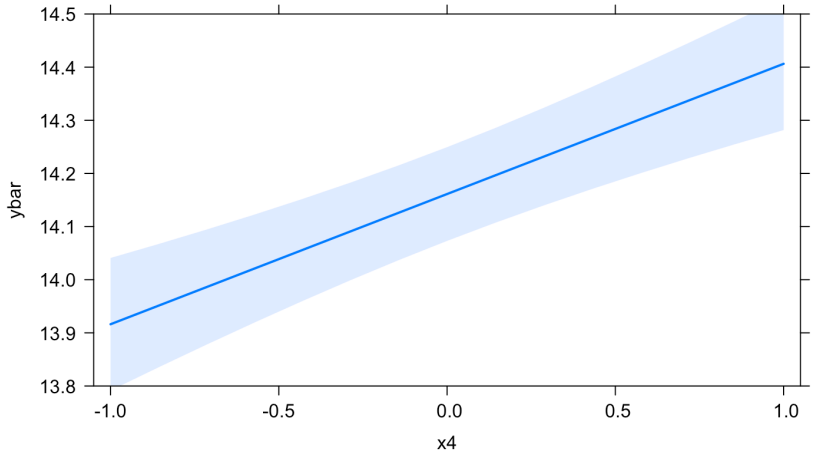
```
library(effects)
plot(Effect("x1", cirfab.lm), main = "", rug = F, ylim = c(13.5, 14.5), aspect = 1)
plot(Effect("x2", cirfab.lm), main = "", rug = F, ylim = c(13.5, 14.5), aspect = 1)
plot(Effect("x3", cirfab.lm), main = "", rug = F, ylim = c(13.5, 14.5), aspect = 1)
plot(Effect("x4", cirfab.lm), main = "", rug = F, ylim = c(13.5, 14.5), aspect = 1)
```

Main effects



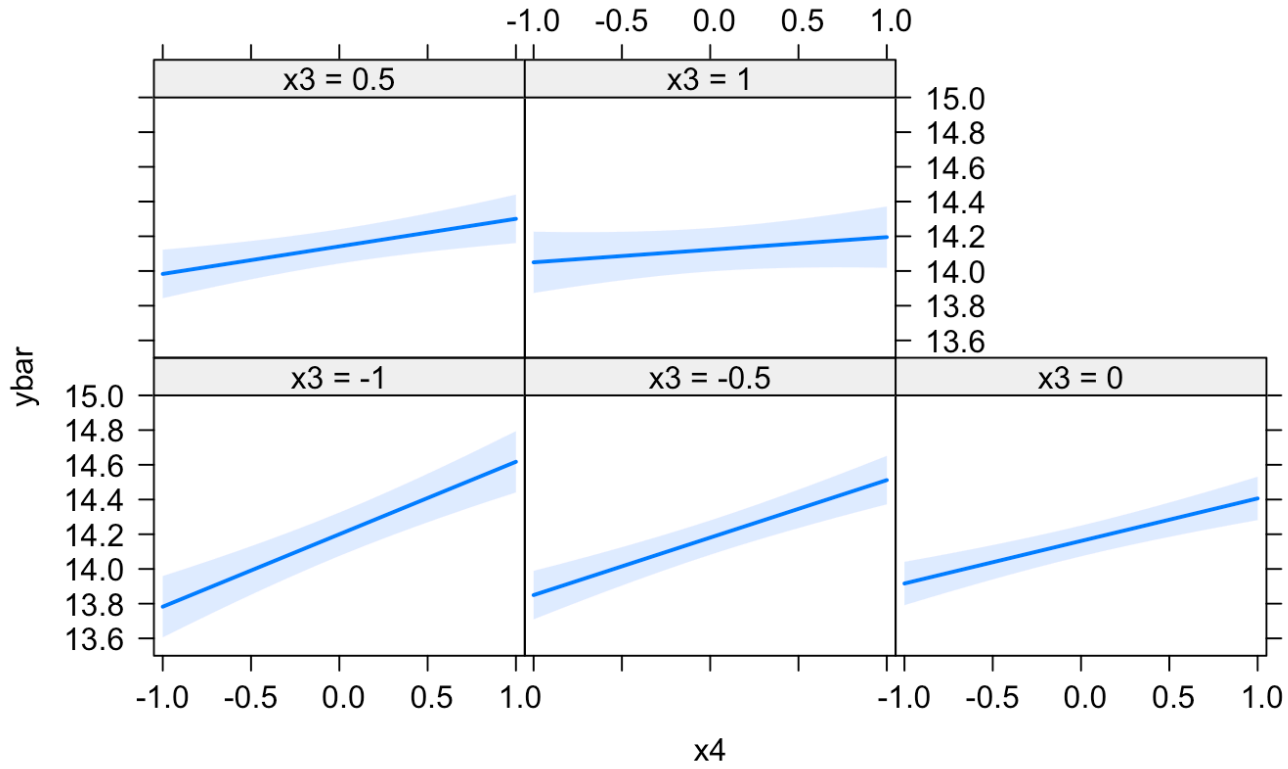






Interactions

```
plot(Effect(c("x3", "x4"), cirfab.lm), main = "", rug = F, ylim = c(13.5, 15),
     x.var = "x4")
```



Orthogonality

$X^{\mathrm{T}}X = nI \Rightarrow \hat{\boldsymbol{\beta}}$ are independently normally distributed with equal variance

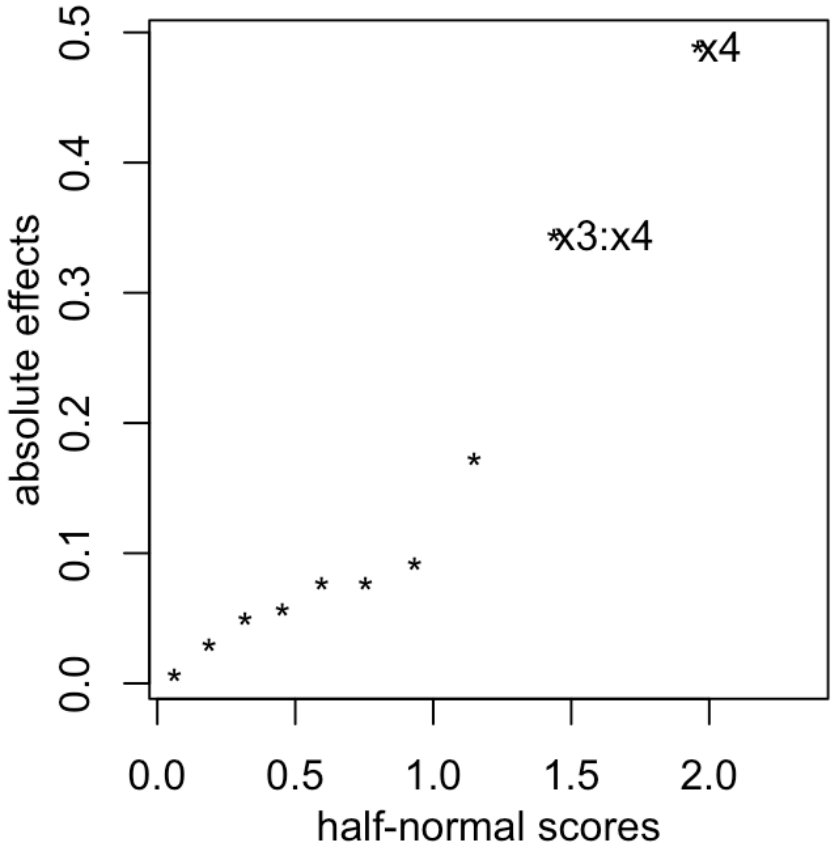
Hence, we can treat the identification of important effects (ie large β) as an outlier identification problem

- plot (absolute) ordered factorial effects against (absolute) quantiles from a standard normal
- outlying effects are identified as important

Cuthbert (1959)

Using the `FrF2` package

```
library(FrF2)
par(pty = "s", mar = c(8, 4, 1, 2))
DanielPlot(cirfab.lm, main = "", datax = F, half = T)
```



Replication

An unreplicated factorial design provides no model-independent estimate of σ^2 (Gilmour and Trinca 2012)

- any unsaturated model does provide an estimate, but it may be biased by ignored (significant) model terms
- this is one reason why graphical (or associated) analysis methods are popular

Replication also increases the power of the design

- common to replicate a centre point
- allows a portmanteau test of curvature

Refit a linear model to the fabricated circuits experiment *excluding* interactions.

```
cirfab2.lm <- lm(ybar ~ (.), data = cirfab)
coef(cirfab.lm)
```

```
##  (Intercept)           x1           x2           x3           x4        x1:x2
## 14.161250000 -0.038729167  0.086270833 -0.038708333  0.245020833  0.003708333
##        x1:x3        x1:x4        x2:x3        x2:x4        x3:x4
## -0.046229167 -0.025000000  0.028770833 -0.015041667 -0.172520833
```

```
coef(cirfab2.lm)
```

```
## (Intercept)          x1          x2          x3          x4
## 14.16125000 -0.03872917  0.08627083 -0.03870833  0.24502083
```

```
cbind(sigma1 = summary(cirfab.lm)$sigma, df1 = summary(cirfab.lm)$df[2],
      sigma2 = summary(cirfab2.lm)$sigma, df2 = summary(cirfab2.lm)$df[2])
```

```
##        sigma1 df1    sigma2 df2
## [1,] 0.137152   5 0.2396108  11
```

Principles of factorial experimentation

Effect sparsity

- the number of important effects in a factorial experiment is small relative to the total number of effects investigated (cf Box and Meyer 1986)

Effect hierarchy

- lower-order effects are more likely to be important than higher-order effects
- effects of the same order are equally likely to be important

Effect heredity

- interactions where at least one parent main effect is important are more likely to be important themselves

Wu and Hamada (2009), pp.172–172

Regular fractional factorial designs

Choosing subsets of treatments

Factorial designs can require a large number of runs for only a moderate number of factors ($2^5 = 32$)

Resource constraints (eg cost) may mean not all 2^m combinations can be run

Lots of degrees of freedom are devoted to estimating higher-order interactions

- eg in a 2^5 experiment, 16 degrees of freedom are used to estimate three-factor and higher-order interactions
- principles of effect hierarchy and sparsity suggest may be wasteful

Need to trade-off **what you want to estimate** against the **number of runs** you can afford

Example

Production of bacteriocin (Morris 2011, p231)

- bacteriocin is a natural food preservative frown from bacteria

Unit

- a single bio-reaction

Treatment: combination of settings of the factors

- A: amount of glucose (x_1)
- B: initial inoculum size (x_2)
- C: level of aeration (x_3)
- D: temperature (x_4)
- E: amount of sodium (x_5)

Assume each factor has two-levels, coded -1 and +1

Find an $n = 8$ run design using FrF2

```
bact.design <- FrF2(8, 5, factor.names = paste0("x", 1:5),
     generators = list(c(1, 3), c(2, 3)), randomize = F, alias.info = 3)
bact.design
```

```
##   x1 x2 x3 x4 x5
## 1 -1 -1 -1  1  1
## 2  1 -1 -1 -1  1
## 3 -1  1 -1  1 -1
## 4  1  1 -1 -1 -1
## 5 -1 -1  1 -1 -1
## 6  1 -1  1  1 -1
## 7 -1  1  1 -1  1
## 8  1  1  1  1  1
## class=design, type= FrF2.generators
```

- $8 = 32/4 = 2^5/2^2 = 2^{5-2}$
- we need a principled way of choosing one-quarter of the runs from the factorial design that leads to clarity in the analysis

Assuming the number of runs is a power of two, $n = 2^{k-q}$, we can construct $2^{k-q} - 1$ orthogonal vectors (with inner product zero), spanned by $k - q = \log_2(n)$ vectors

- construct the full factorial design for $k - q$ factors
- assign the remaining q factors to interaction columns

```
model.matrix(~ (x1 + x2 + x3) ^ 3, bact.design[, 1:3])[, ]
```

```
##   (Intercept) x11 x21 x31 x11:x21 x11:x31 x21:x31 x11:x21:x31
## 1           1  -1  -1  -1       1       1       1          -1
## 2           1   1  -1  -1      -1      -1       1           1
## 3           1  -1   1  -1      -1       1      -1           1
## 4           1   1   1  -1       1      -1      -1          -1
## 5           1  -1  -1   1       1      -1      -1           1
## 6           1   1  -1   1      -1       1      -1          -1
## 7           1  -1   1   1      -1      -1       1          -1
## 8           1   1   1   1       1       1       1           1
```

Aliasing scheme

The design has been deliberately chosen so that

- $x_4 = x_1 x_3$
- $x_5 = x_2 x_3$

[$x_1 x_2$ is shorthand for the Hadamard (Schur or entry wise) product of two vectors, $x_1 \circ x_2$]

What other consequences are there?

- $x_4 x_5 = x_1 x_3 x_2 x_3 = x_1 x_2 x_3^2$
- the product of any column with itself is the constant column (the identity)
- hence, $x_4 x_5 = x_1 x_2$

Now we can obtain the **defining relation** …

- $I = x_1 x_3 x_4 = x_2 x_3 x_5 = x_1 x_2 x_4 x_5$

… and the complete aliasing scheme

- $x_1 = x_3 x_4 = x_1 x_2 x_3 x_5 = x_2 x_4 x_5$
- $x_2 = x_1 x_2 x_3 x_4 = x_3 x_5 = x_1 x_4 x_5$
- $x_3 = x_1 x_4 = x_2 x_5 = x_1 x_2 x_3 x_4 x_5$
- $x_4 = x_1 x_3 = x_2 x_3 x_4 x_5 = x_1 x_2 x_5$
- $x_5 = x_1 x_3 x_4 x_5 = x_2 x_3 = x_1 x_2 x_4$
- $x_1 x_2 = x_2 x_3 x_4 = x_1 x_3 x_5 = x_4 x_5$
- $x_1 x_5 = x_3 x_4 x_5 = x_1 x_2 x_3 = x_2 x_4$

`FrF2` will summarise the aliasing amongst main effects and two- and three-factor interactions.

```
design.info(bact.design)$aliased
```

```
## $legend
## [1] "A=x1" "B=x2" "C=x3" "D=x4" "E=x5"
##
## $main
## [1] "A=CD=BDE" "B=CE=ADE" "C=AD=BE"  "D=AC=ABE" "E=BC=ABD"
##
## $fi2
## [1] "AB=DE=ACE=BCD" "AE=BD=ABC=CDE"
##
## $fi3
## [1] "ACD=BCE"
```

The alias matrix

What is the consequence of this aliasing?

If more than one effect in each alias string is non-zero, the least squares estimators will be biased

- assumed model $\boldsymbol{Y} = X_1\boldsymbol{\beta}_1 + \boldsymbol{\varepsilon}$
- true model $\boldsymbol{Y} = X_1\boldsymbol{\beta}_1 + X_2\boldsymbol{\beta}_2 + \boldsymbol{\varepsilon}$

$$\begin{aligned} E\left(\hat{\boldsymbol{\beta}}_1\right) &= \left(X_1^{\mathrm{T}}X_1\right)^{-1}X_1^{\mathrm{T}}E(\boldsymbol{Y}) \\ &= \left(X_1^{\mathrm{T}}X_1\right)^{-1}X_1^{\mathrm{T}}\left(X_1\boldsymbol{\beta}_1 + X_2\boldsymbol{\beta}_2\right) \\ &= \beta_1 + \left(X_1^{\mathrm{T}}X_1\right)^{-1}X_1^{\mathrm{T}}X_2\boldsymbol{\beta}_2 \\ &= \boldsymbol{\beta}_1 + A\boldsymbol{\beta}_2 \end{aligned}$$

A is the alias matrix

- if the columns of X_1 and X_2 are not orthogonal, $\hat{\boldsymbol{\beta}}_1$ is biased

For the 2^{5-2} example:

- X_1 is an 8×6 matrix with columns for the intercept and five linear terms ("main effects")
- X_2 is an 8×10 matrix with columns for the 10 product terms ("two-factor interactions")

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For a regular design, the matrix A will only have entries 0, ± 1 (no aliasing or complete aliasing)

The **transpose** of the alias matrix is provided by the `alias` function.

```
ff.alias <- alias(y ~ (.)^2, data = data.frame(bact.design, y = vector(length = 8)))
ff.alias$Complete
```

```
##         (Intercept) x11 x21 x31 x41 x51 x11:x21 x11:x51
## x21:x31 0           0   0   0   0   1   0       0
## x21:x41 0           0   0   0   0   0   0       1
## x21:x51 0           0   0   1   0   0   0       0
## x31:x41 0           1   0   0   0   0   0       0
## x31:x51 0           0   1   0   0   0   0       0
## x41:x51 0           0   0   0   0   0   1       0
## x11:x31 0           0   0   0   1   0   0       0
## x11:x41 0           0   0   1   0   0   0       0
```

These linear dependencies can be seen if we attempt to fit a linear model

```
bact.lm <- lm(yB ~ (x1 + x2 + x3 + x4 + x5)^2, data = bact)
summary(bact.lm)
```

```
##
## Call:
## lm.default(formula = yB ~ (x1 + x2 + x3 + x4 + x5)^2, data = bact)
##
## Residuals:
## ALL 8 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (8 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.42625         NA      NA       NA
## x1           0.26625         NA      NA       NA
## x2           0.24625         NA      NA       NA
## x3          -0.22875         NA      NA       NA
## x4          -1.11875         NA      NA       NA
## x5          -0.66375         NA      NA       NA
## x1:x2       -0.05375         NA      NA       NA
## x1:x3             NA         NA      NA       NA
## x1:x4             NA         NA      NA       NA
## x1:x5       -0.13375         NA      NA       NA
## x2:x3             NA         NA      NA       NA
## x2:x4             NA         NA      NA       NA
## x2:x5             NA         NA      NA       NA
## x3:x4             NA         NA      NA       NA
## x3:x5             NA         NA      NA       NA
## x4:x5             NA         NA      NA       NA
```

The role of fractional factorial designs in a sequential strategy

Typically, in a first experiment, fractional factorial designs are used in screening

- investigate which of many factors have a substantive effect on the response
- main effects and two-factor interactions
- centre points to check for curvature

At second and later stages, augment the design

- to resolve ambiguities due to the aliasing of factorial effects ("break the alias strings")
- to allow estimation of curvature and prediction from a more complex model

D-optimality and non-regular designs

Introduction

Regular fractional factorial designs have the number of runs equal to a power of the number of levels

- eg 2^{5-2}, $3^{3-1} \times 2$
- this inflexibility in run sizes can be a problem in practical experiments

Non-regular designs can have any number of runs (usually with $n > p$, the number of parameters to be estimated)

Often the clarity provided by a regular design is lost

- no defining relation or straightforward aliasing scheme
- partial aliasing and fractional entries in A

One approach to finding non-regular designs is via a design optimality criterion

D-optimality

Notation: let $\xi = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ denote a design (choice of treatments and their replications)

Assuming the model $\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, with $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I_n)$, a D**-optimal** design maximises

$$\phi(\xi) = \det\left(X^{\mathrm{T}}X\right)$$

That is, a D-optimal design maximises the determinant of the (expected) Fisher information matrix

- equivalent to minimising the volume of the joint confidence ellipsoid for $\boldsymbol{\beta}$

Also useful to define a Bayesian version, with R a prior precision matrix

$$\phi_B(\xi) = \det\left(X^{\mathrm{T}}X + R\right)$$

(See later)

Comments

D-optimal designs are model dependent

- if the model (ie the columns of X) changes, the optimal design may change
- model-robust design is an active area of research

D-optimality promotes orthogonality in the X matrix

- if there are sufficient runs, the D-optimal design will usually be orthogonal
- for particular models and choices of n, regular fractional factorial designs are D-optimal

There are many other optimality criteria, tailored to other experimental goals

- prediction, model discrimination, space-filling, …

Example: Plackett-Burman design

$k = 11$ factors in $n = 12$ runs, first-order (main effects) model (Plackett and Burman 1946)

A particular D-optimal design is the following orthogonal array

Using the `pb` function in the `FrF2` package:

```
pb.design <- pb(12, factor.names = paste0("x", 1:11))
pb.design
```

```
##    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11
## 1  -1  1 -1  1  1 -1  1  1  1  -1  -1
## 2  -1 -1  1 -1  1  1 -1  1  1   1  -1
## 3  -1 -1 -1  1 -1  1  1 -1  1   1   1
## 4   1  1  1 -1 -1 -1  1 -1  1   1  -1
## 5  -1  1  1 -1  1  1  1 -1 -1  -1   1
## 6   1 -1 -1 -1  1 -1  1  1 -1   1   1
## 7  -1  1  1  1 -1 -1 -1  1 -1   1   1
## 8  -1 -1 -1 -1 -1 -1 -1 -1 -1  -1  -1
## 9   1 -1  1  1 -1  1  1  1 -1  -1  -1
## 10  1  1 -1  1  1  1 -1 -1 -1   1  -1
## 11  1 -1  1  1  1 -1 -1 -1  1  -1   1
## 12  1  1 -1 -1 -1  1 -1  1  1  -1   1
## class=design, type= pb
```

This 12-run PB design is probably the most studied non-regular design

- orthogonal columns
- complex aliasing between main effects and two-factor interactions

```
pb.alias <- alias(y ~ (.)^2, data = data.frame(pb.design, y = vector(length = 12)))
head(pb.alias$Complete, n = 15)
```

```
##          (Intercept) x11  x21  x31  x41  x51  x61  x71  x81  x91  x101 x111
## x11:x21     0           0    0 -1/3 -1/3 -1/3  1/3 -1/3 -1/3  1/3  1/3 -1/3
## x11:x31     0           0 -1/3    0  1/3 -1/3 -1/3  1/3 -1/3  1/3 -1/3 -1/3
## x11:x41     0           0 -1/3  1/3    0  1/3  1/3 -1/3 -1/3 -1/3 -1/3 -1/3
## x11:x51     0           0 -1/3 -1/3  1/3    0 -1/3 -1/3 -1/3 -1/3  1/3  1/3
## x11:x61     0           0  1/3 -1/3  1/3 -1/3    0 -1/3  1/3 -1/3 -1/3 -1/3
## x11:x71     0           0 -1/3  1/3 -1/3 -1/3 -1/3    0  1/3 -1/3  1/3 -1/3
## x11:x81     0           0 -1/3 -1/3 -1/3 -1/3  1/3  1/3    0 -1/3 -1/3  1/3
## x11:x91     0           0  1/3  1/3 -1/3 -1/3 -1/3 -1/3 -1/3    0 -1/3  1/3
## x11:x101    0           0  1/3 -1/3 -1/3  1/3 -1/3  1/3 -1/3 -1/3    0 -1/3
## x11:x111    0           0 -1/3 -1/3 -1/3  1/3 -1/3 -1/3  1/3  1/3 -1/3    0
## x21:x31     0        -1/3    0    0 -1/3 -1/3 -1/3  1/3 -1/3 -1/3  1/3  1/3
## x21:x41     0        -1/3    0 -1/3    0  1/3 -1/3 -1/3  1/3 -1/3  1/3 -1/3
## x21:x51     0        -1/3    0 -1/3  1/3    0  1/3  1/3 -1/3 -1/3 -1/3 -1/3
## x21:x61     0         1/3    0 -1/3 -1/3  1/3    0 -1/3 -1/3 -1/3 -1/3  1/3
## x21:x71     0        -1/3    0  1/3 -1/3  1/3 -1/3    0 -1/3  1/3 -1/3 -1/3
```

Example: supersaturated design

Screening designs with fewer runs than factors (see Woods and Lewis 2017)

- can't use ordinary least squares/maximum likelihood as X does not have full column rank
- Bayesian D-optimality with $R = [0 \mid \tau I_m]$

Supersaturated experiment used by GlaxoSmithKline in the development of a new oncology drug

- $k = 16$ factors: e.g. temperature, solvent amount, reaction time
- $n = 10$ runs
- Bayesian D-optimal design with $\tau = 0.2$

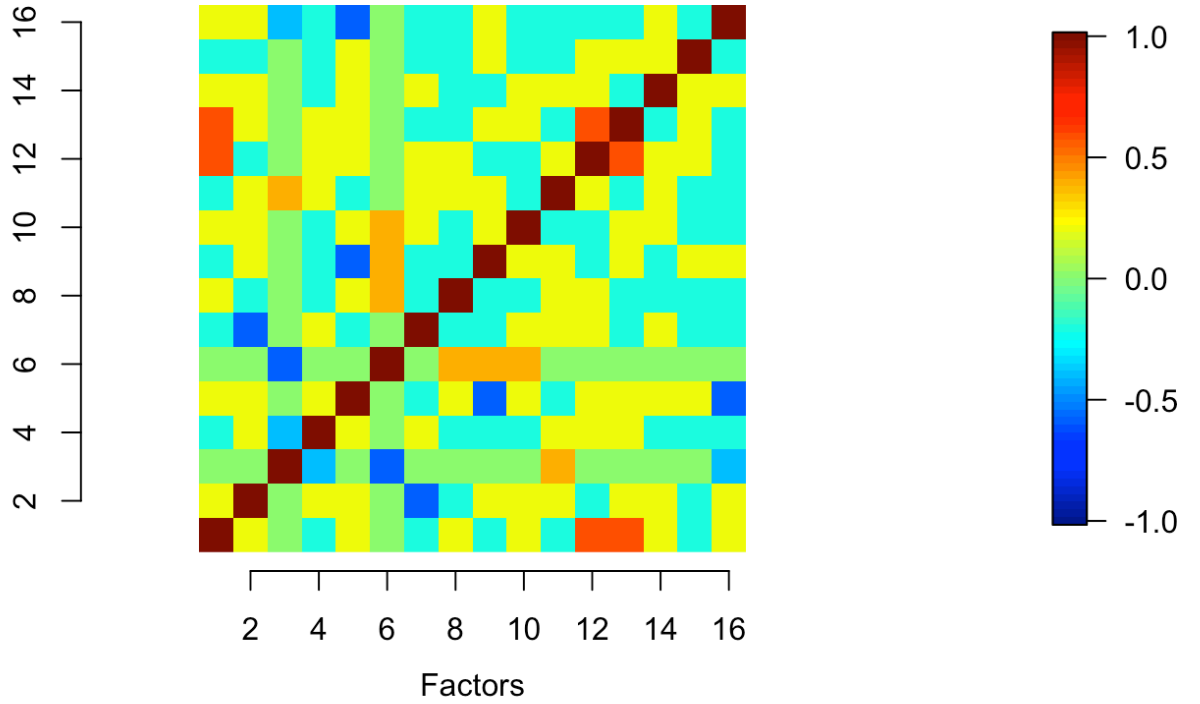
```
ssd
```

```
##    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16
## 1   1  1 -1  1  1 -1 -1 -1 -1  -1  -1   1   1   1   1   1
## 2   1  1  1 -1 -1 -1 -1 -1  1   1  -1  -1   1  -1  -1   1
## 3  -1 -1  1 -1  1 -1  1 -1 -1   1  -1  -1  -1   1   1  -1
## 4  -1  1  1  1  1 -1 -1  1 -1  -1   1  -1  -1  -1  -1  -1
## 5  -1 -1 -1 -1 -1  1 -1  1  1  -1  -1  -1  -1  -1   1   1
## 6   1  1  1 -1  1  1 -1  1  1   1   1   1   1   1   1  -1
## 7  -1 -1  1  1 -1 -1  1 -1  1  -1   1   1   1  -1   1  -1
## 8   1 -1 -1  1  1  1  1  1 -1   1  -1   1   1  -1  -1  -1
## 9  -1  1 -1  1 -1  1  1 -1  1   1   1  -1  -1   1  -1   1
## 10  1 -1  1 -1 -1 -1  1  1 -1  -1   1   1  -1   1  -1   1
```

Partial aliasing between main effects

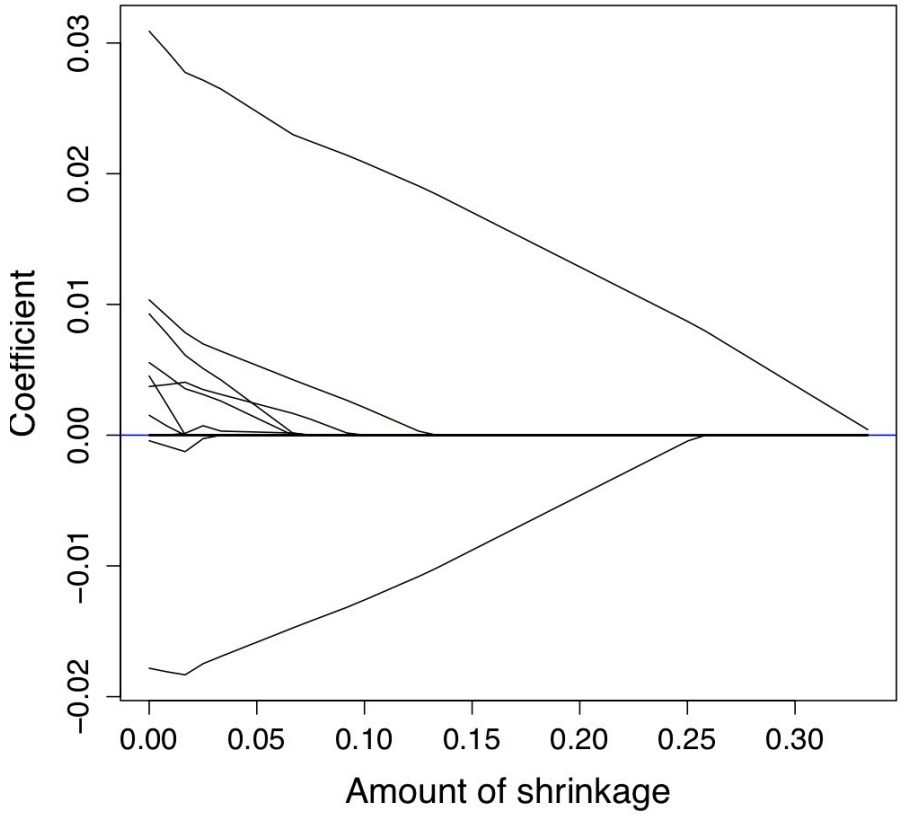
Heatmap of column correlations:

```
library(fields)
par(mar=c(8,2,0,0))
image.plot(1:16,1:16, cor(ssd), zlim = c(-1, 1), xlab = "Factors",
           ylab = "", asp = 1, axes = F)
axis(1, at = seq(2, 16, by = 2), line = .5)
axis(2, at = seq(2, 16, by = 2), line = -5)
```



Analysis via regularised (shrinkage) methods (eg lasso, Dantzig selector; see APTS High Dimensional Statistics)

- small coefficients shrunk to zero



Bayesian optimal design

Introduction

Now consider a more general class of models (cf preliminary material).

Let $\boldsymbol{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ be iid observations from a distribution with density/mass function $\pi(y_i\,;\,\boldsymbol{\theta}, \boldsymbol{x}_i)$

- $\boldsymbol{\theta}$ is a $q-$vector of unknown parameters
- $\boldsymbol{x}_i = (x_{1i}, \ldots, x_{ki})^{\mathrm{T}}$ is a vector of values of k controllable variables.

The (expected) information matrix

$$M(\boldsymbol{\theta}) = E_y\left[-\frac{\partial^2 l(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{\mathrm{T}}}\right]$$

is an important quantity for design, where $l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \pi(y_i;\,\boldsymbol{\theta}, \boldsymbol{x}_i)$ (the log-likelihood).

- $M(\boldsymbol{\theta})$ is the (asymptotic) precision for the maximum likelihood estimators $\hat{\boldsymbol{\theta}}$.
- $M(\boldsymbol{\theta})$ is also an asymptotic approximation to the posterior precision for $\boldsymbol{\theta}$ in a Bayesian analysis.

Pharmacokinetics

Example 1: Compartmental model

$$y_i \sim N\left(c(\boldsymbol{\theta})\mu(\boldsymbol{\theta};\, x_i), \sigma^2\nu(\boldsymbol{\theta};\, x_i)\right)\,, \qquad x_i \in [0, 24]\,,$$

with

$$\mu(\boldsymbol{\theta};\, x) = \exp(-\theta_1 x) - exp(-\theta_2 x)\,, \qquad c(\boldsymbol{\theta}) = \frac{400\theta_2}{\theta_3(\theta_2 - \theta_1)}\,, \qquad \nu(\boldsymbol{\theta};\, x) = 1 + \frac{\tau^2}{\sigma^2}c(\boldsymbol{\theta})^2\mu(\boldsymbol{\theta};\, x)\,,$$

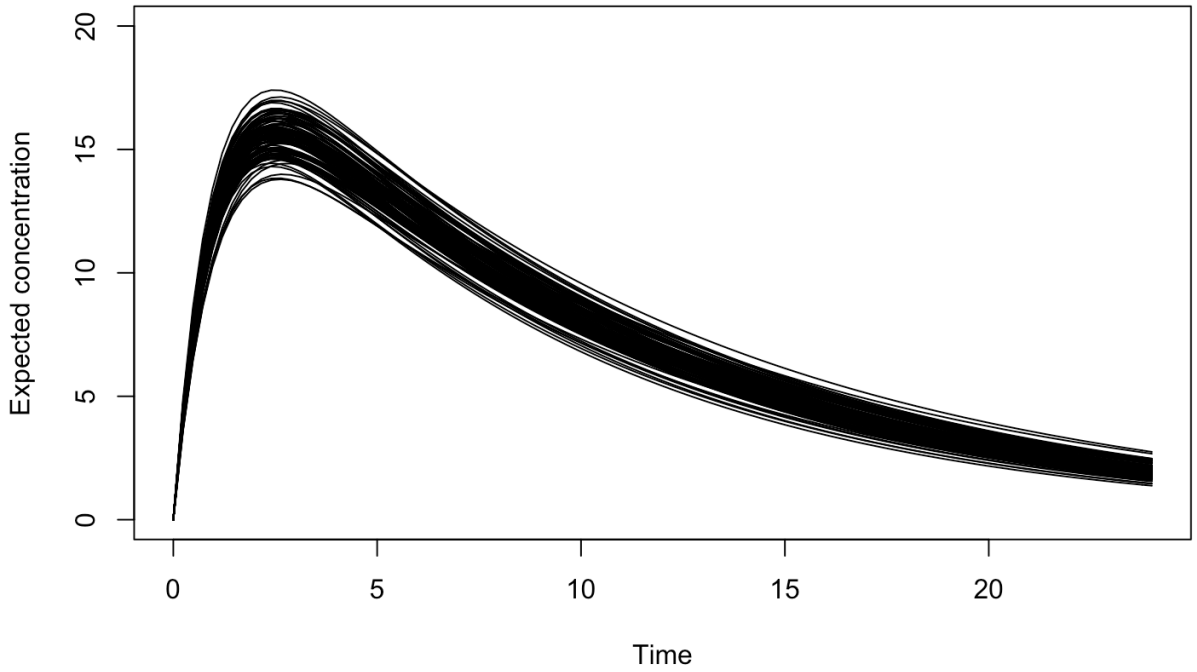
for $\theta_1, \theta_2, \theta_3, \tau^2, \sigma^2 > 0$.

Prior distributions (for later use):

- $\log\theta_i \sim N(m_i, 0.05)$, with $m_1 = \log 0.1, m_2 = 0, m_3 = \log 20$

Ryan et al. (2014)

```
comp <- function(x, theta, D = 400) {
    mu <- exp(-theta[1] * x) - exp(-theta[2] * x)
    c <- (D / theta[3]) * (theta[2]) / (theta[2] - theta[1])
    c * mu }
theta <- c(.1, 1, 20)
M <- 100
par(mar = c(6, 4, 0, 1) + .1)
lapply(1:M, function(l) {
  thetat <- rlnorm(3,log(theta),rep(0.05,3))
  curve(comp(x, theta = thetat), from = 0, to = 24, ylab = "Expected concentration",
        xlab = "Time", ylim = c(0, 20), xlim = c(0, 24), add = l!=1) })
```

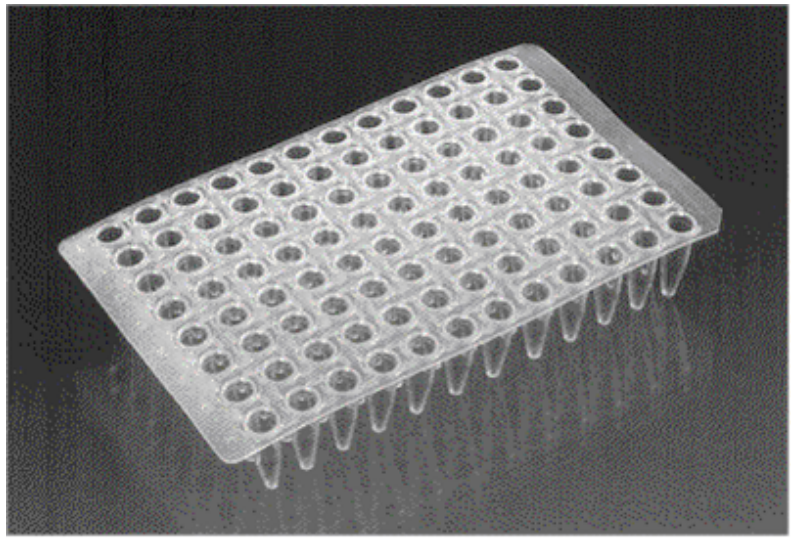


Crystallography

Example 2: multi-factor experiments to build (hierarchical) logistic regression models for pharmaceutical salt formation

Four controllable variables:

- rate of agitation during mixing (x_1)
- volume of composition (x_2)
- temperature (x_3)
- evaporation rate (x_4)



For the jth observation in the ith group ($i = 1, \ldots, g$; $j = 1, \ldots, n_g$):

$$y_{ij} \sim \text{Bernoulli}\left(\rho(\boldsymbol{x}_{ij})\right)$$

with

$$\log\left(\frac{\rho(\boldsymbol{x}_{ij})}{1-\rho(\boldsymbol{x}_{ij})}\right) = (\beta_0 + \omega_{i0}) + \sum_{r=1}^{k} (\beta_r + \omega_{ir})\, x_{ijr}\,,$$

where x_{ijr} is the value taken by the rth variable.

- $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_{q-1})^{\mathrm{T}}$ are unknown parameters of interest
- $\boldsymbol{\omega}_i = (\omega_{i0}, \omega_{i1}, \ldots, \omega_{iq-1})^{\mathrm{T}}$ are group specific parameters for the ith group

Prior distributions (for later use):

- $\beta_0 \sim U(-3, 3)$, $\beta_1 \sim U(4, 10)$, $\beta_2 \sim U(5, 11)$, $\beta_3 \sim U(-6, 0)$, $\beta_4 \sim U(-2.5, 3.5)$

1. standard logistic regression - $\omega_{ir} = 0$
2. hierarchical logistic regression - $\omega_{ir} \sim U(-s_r, +s_r)$. with $s_r > 0$ following a triangular distribution

Classical optimal designs

Many Frequentist criteria for finding optimal designs for both linear and nonlinear models optimise a function of the information matrix; see Atkinson, Donev, and Tobias (2007), ch.10

- we have already seen D-optimality

Let $\xi = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)^{\mathrm{T}}$ denote a design, and set $M(\xi; \boldsymbol{\theta}) = M(\boldsymbol{\theta})$ to explicitly acknowledge the dependence of the information matrix on the design

- D**-optimality**: maximise $\phi_D(\xi) = \det M(\xi; \boldsymbol{\theta})$
- A**-optimality**: minimise $\phi_A(\xi) = \operatorname{trace} M(\xi; \boldsymbol{\theta})^{-1}$
- G**-optimality**: minimise $\phi_G(\xi) = \max_{\boldsymbol{x}} \operatorname{Var}(\hat{y}(\boldsymbol{x}))$
  - where $\hat{y}(x)$ is the predicted response at $\boldsymbol{x}$ and the (asymptotic) prediction variance is a function of $M(\xi; \boldsymbol{\theta})$
- V**- (or** I**-) optimality** - minimise $\phi_V(\xi) = \int_{\mathcal{X}} \operatorname{Var}\left(\hat{y}(\boldsymbol{x})\right) \, \mathrm{d}\boldsymbol{x}$

Optimal design for nonlinear models

For most nonlinear models, $M(\xi; \boldsymbol{\theta})$ will be a function of the unknown parameters $\boldsymbol{\theta}$ (unlike for the linear model, where $M(\xi; \boldsymbol{\beta}) = X^{\mathrm{T}}X$)

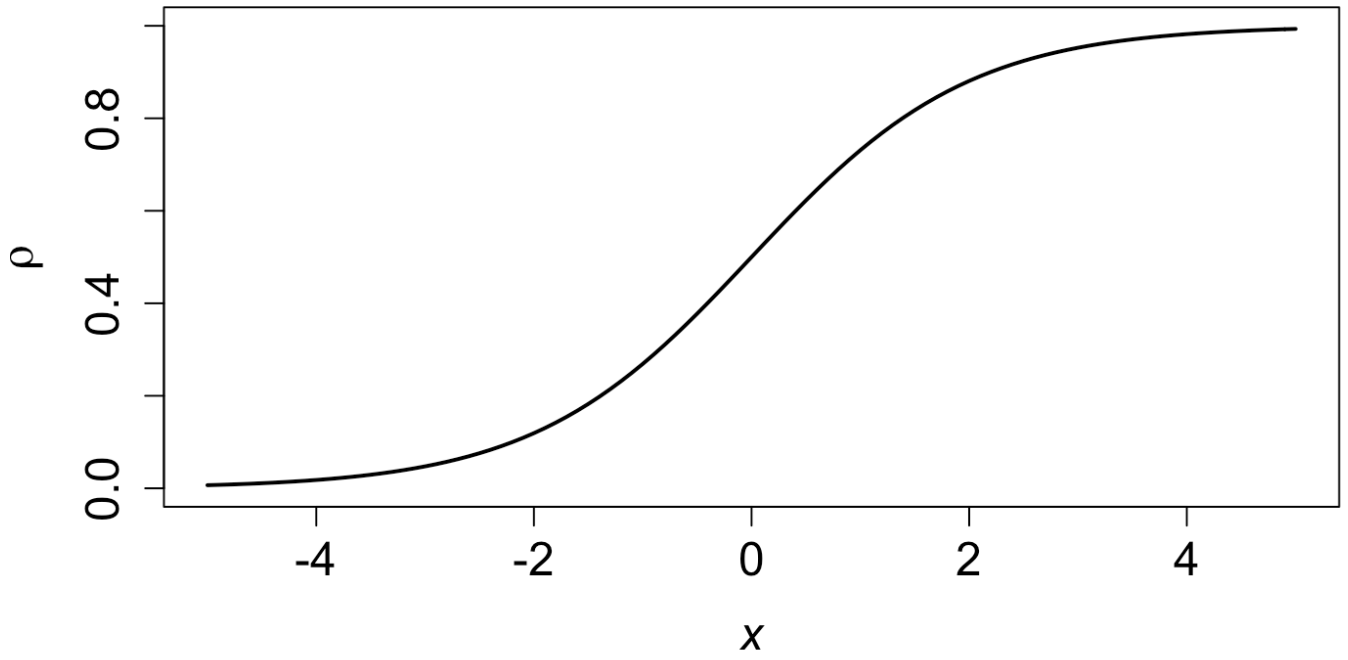
This leads to a "chicken and egg" situation

- if you can tell me the values of the unknown parameters, I can give you an optimal design
- but if you knew the value of $\boldsymbol{\theta}$, you probably wouldn't need to perform the experiment!

For some models/experiments, the quality of a design may change a lot with the value of $\boldsymbol{\theta}$

A simple example

```
rho <- function(x, beta0 = 0, beta1 = 1) {
  eta <- beta0 + beta1 * x
  1 / (1 + exp(-eta))
}
par(mar = c(8, 4, 1, 2) + 0.1)
curve(rho, from = -5, to = 5, ylab = expression(rho), xlab = expression(italic(x)), cex.lab = 1.5,
      cex.axis = 1.5, ylim = c(0, 1), lwd = 2)
```



For simple logistic regression, the information matrix has the form

$$M(\xi; \boldsymbol{\beta}) = X^{\mathrm{T}} W X \,,$$

with X the $n \times 2$ model matrix and $W = \mathrm{diag}\,\{\rho(x_i)[1 - \rho(x_i)]\}$

For example with $n = 2$, $\xi = (-1, 1)$, $\beta_0 = 0$ and $\beta_1 = 1$

$$M(\xi; \boldsymbol{\beta}) = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

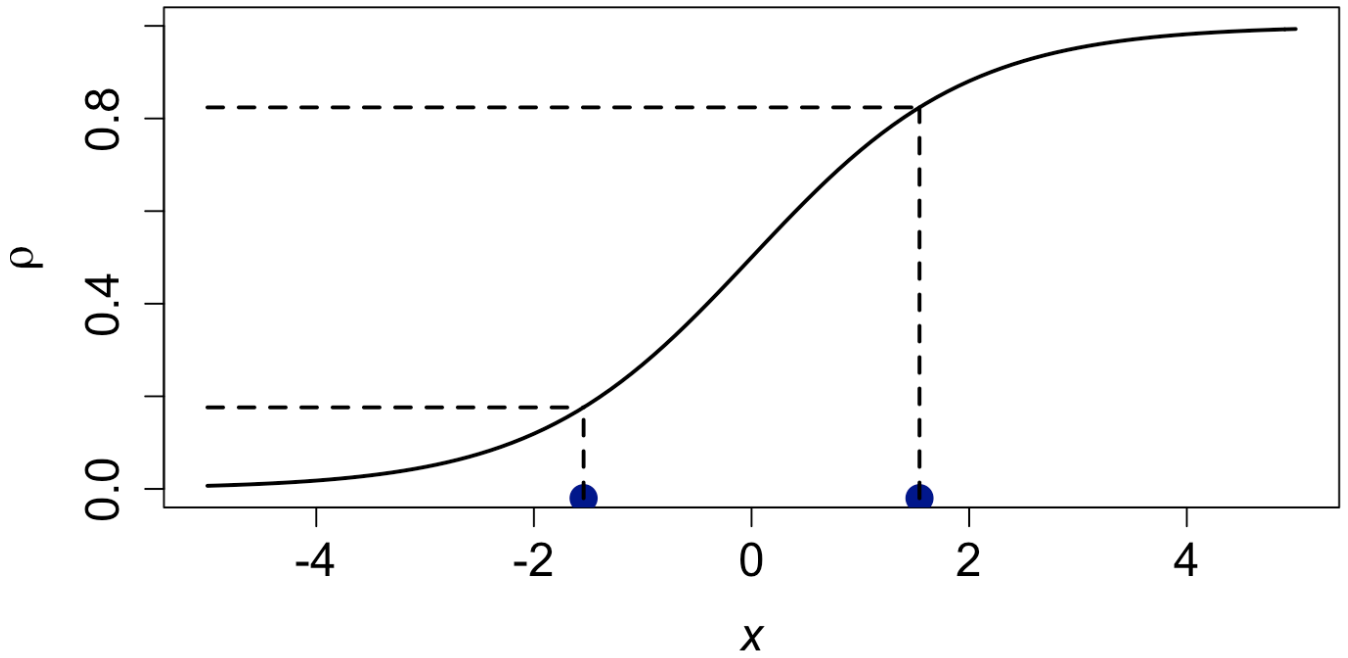
```
Minfo <- function(xi, beta0 = 0, beta1 = 1) {
  X <- cbind(c(1, 1), xi)
  v <- function(x) rho(x, beta0, beta1) * (1 - rho(x, beta0, beta1))
  W <- diag(c(v(xi[1]), v(xi[2])))
  t(X) %*% W %*% X
}
Dcrit <- function(xi, beta0 = 0, beta1 = 1) {
  d <- det(Minfo(xi, beta0, beta1))
  ifelse(is.nan(d), -Inf, d)
}
```

Locally D-optimal designs

$\beta_0 = 0, \beta_1 = 1$

```
dopt <- optim(par = c(-1, 1), Dcrit, control = list(fnscale = -1))
xi.opt1 <- dopt$par
xi.opt1
```

```
## [1] -1.543421  1.543530
```

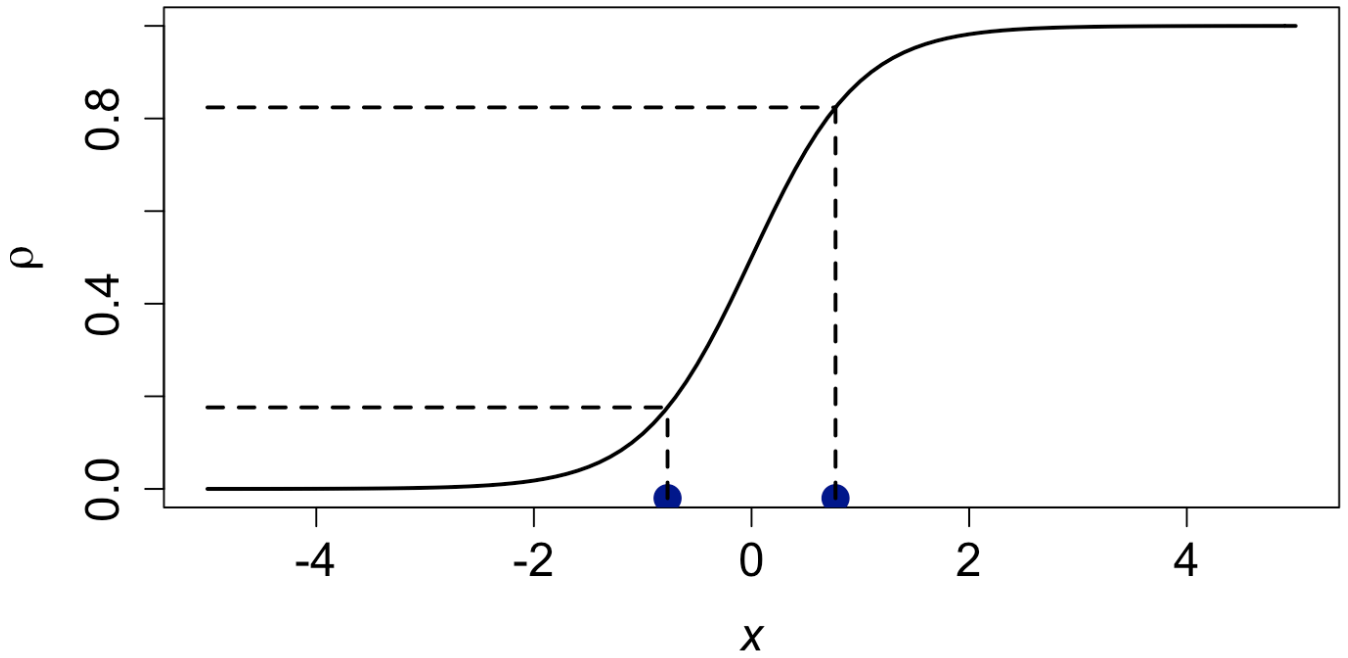


Locally D-optimal designs

$\beta_0 = 0, \beta_1 = 2$

```
dopt <- optim(par = c(-1, 1), Dcrit, control = list(fnscale = -1), beta1 = 2)
xi.opt2 <- dopt$par
xi.opt2
```

```
## [1] -0.7717705  0.7717418
```

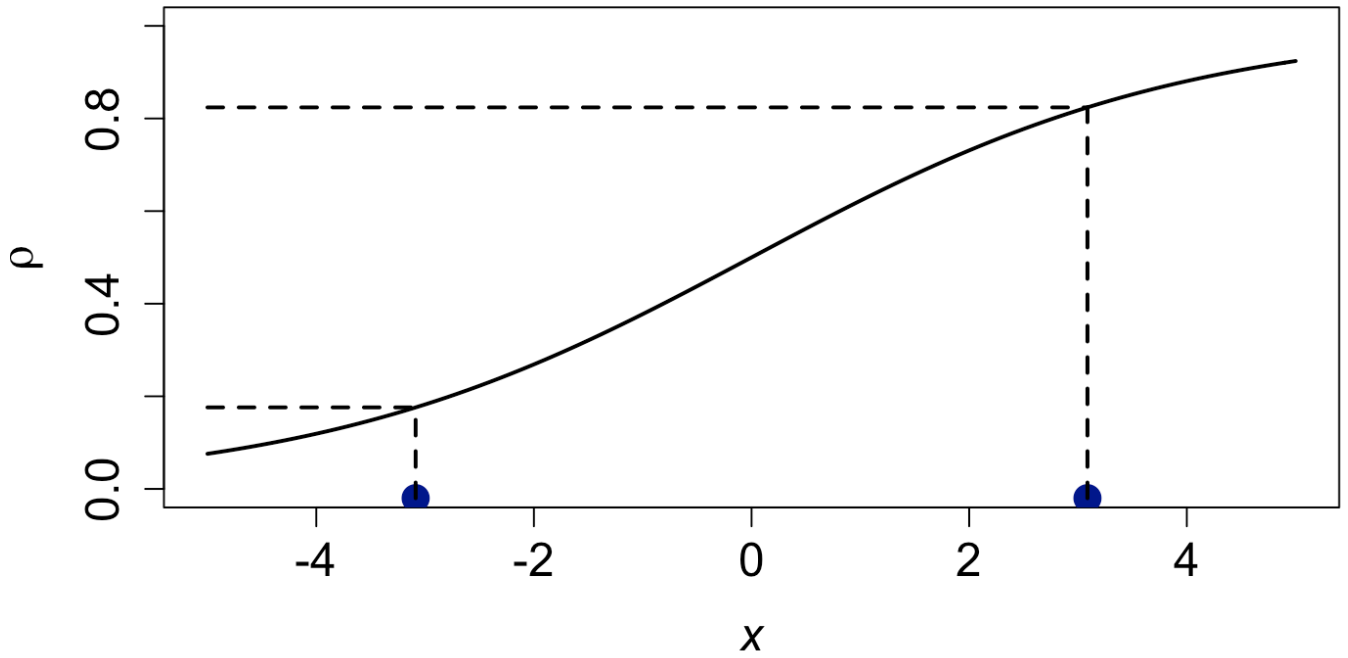


Locally D-optimal designs

$\beta_0 = 0, \beta_1 = 0.5$

```
dopt <- optim(par = c(-1, 1), Dcrit, control = list(fnscale = -1), beta1 = .5)
xi.opt3 <- dopt$par
xi.opt3
```

```
## [1] -3.086913  3.086981
```

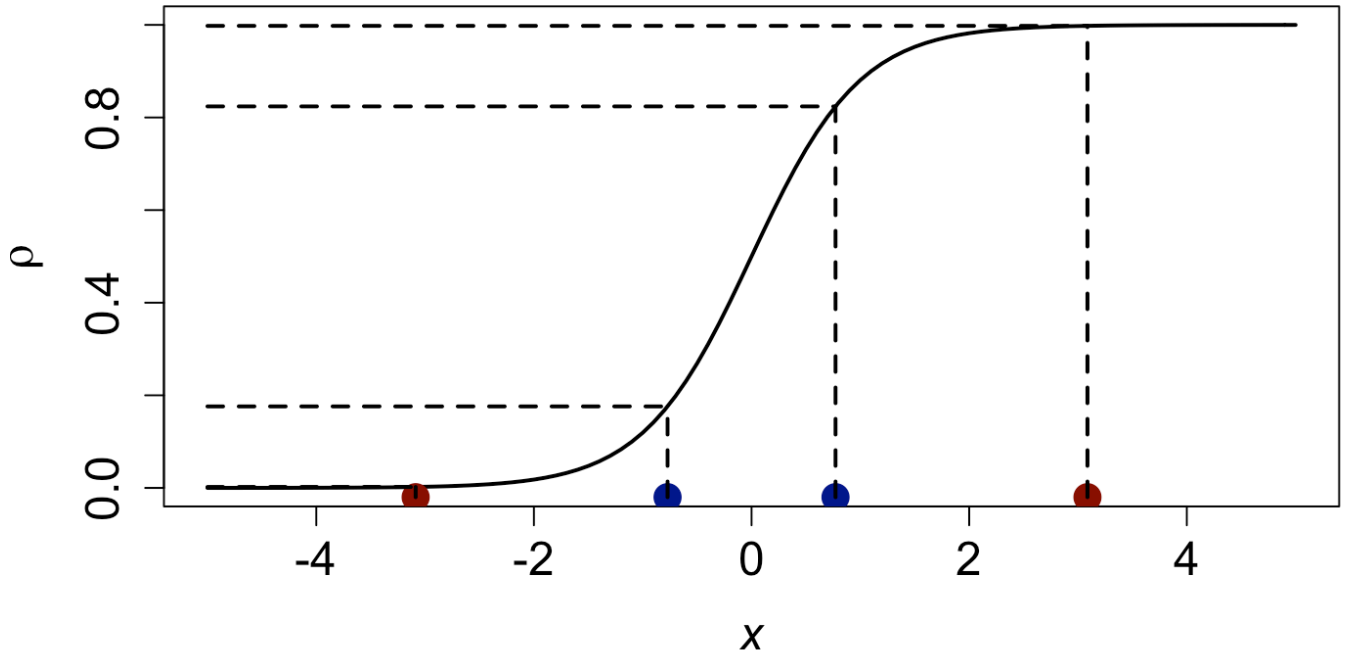


Getting β_1 wrong: design for $\beta_1 = .5$ when actually $\beta_1 = 2$

D-efficiency

```
(Dcrit(xi.opt3, beta1 = 2) / Dcrit(xi.opt2, beta1 = 2)) ^ (1 / 2)
```

```
## [1] 0.05720905
```



Use of the "wrong" design can lead to uninformative experiments (with "small" information matrices)

For the logistic regression example, the drop in efficiency is closely related to the phenomenon of separation (see Firth 1993)

Motivates the need for designs which are robust to the values of the model parameters

- maximin designs (focus on worst case performance)
- **Bayesian designs**

Bayesian optimal design

Decision-theoretic design starts with a utility function $u(\xi, \boldsymbol{y}, \boldsymbol{\theta})$ that defines the usefulness of a design for a particular purpose, given data $\boldsymbol{y}$ and parameters $\boldsymbol{\theta}$

Common choices of utility function include

- negative squared error loss

$$u(\xi, \boldsymbol{y}, \boldsymbol{\theta}) = -[\boldsymbol{\theta} - E(\boldsymbol{\theta} \mid \boldsymbol{y})]^2$$

  - negative squared difference between $\boldsymbol{\theta}$ and the posterior mean

- surprisal or self information

$$\begin{aligned} u(\xi, \boldsymbol{y}, \boldsymbol{\theta}) &= \log \pi(\boldsymbol{\theta} \mid \boldsymbol{y}, \xi) - \log \pi(\boldsymbol{\theta}) \\ &= \log \pi(\boldsymbol{y} \mid \boldsymbol{\theta}, \xi) - \log \pi(\boldsymbol{y} \mid \xi) \end{aligned}$$

  - difference between log posterior and log prior densities, or between the log-likelihood and the log-evidence

A priori (before the experiment), we do not know $\boldsymbol{y}$ or $\boldsymbol{\theta}$ (we will never **know** $\boldsymbol{\theta}$)

So, we take the expectation of the utility function with respect to the joint distribution of $\boldsymbol{y}, \boldsymbol{\theta}$

$$
\begin{aligned}
U(\xi) &= E_{\boldsymbol{y},\boldsymbol{\theta} \mid \xi}\left[u(\xi, \boldsymbol{y}, \boldsymbol{\theta})\right] \\
&= \int u(\xi, \boldsymbol{y}, \boldsymbol{\theta}) \pi(\boldsymbol{y}, \boldsymbol{\theta} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y} \\
&= \int u(\xi, \boldsymbol{y}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta} \mid \boldsymbol{y}, \xi) \pi(\boldsymbol{y} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y} \\
&= \int u(\xi, \boldsymbol{y}, \boldsymbol{\theta}) \pi(\boldsymbol{y} \mid \boldsymbol{\theta}, \xi) \pi(\boldsymbol{\theta} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y}
\end{aligned}
$$

The equivalence of the third and fourth equations follows from Bayes theorem

- the third equation more clearly shows the dependence on the posterior distribution
- the fourth equation is often more useful for calculations and computation

See Chaloner and Verdinelli (1995)

Surprisal

$$U(\xi) = \int \log \frac{\pi(\boldsymbol{\theta} \mid \boldsymbol{y}, \xi)}{\pi(\boldsymbol{\theta})} \pi(\boldsymbol{y}, \boldsymbol{\theta} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y}$$
$$= \int \log \frac{\pi(\boldsymbol{y} \mid \boldsymbol{\theta}, \xi)}{\pi(\boldsymbol{y} \mid \xi)} \pi(\boldsymbol{y}, \boldsymbol{\theta} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y}$$

- the **expected Shannon information gain (SIG)** or expected Kullback-Liebler divergence between prior and posterior densities

Negative squared error loss

$$U(\xi) = -\int [\boldsymbol{\theta} - E(\boldsymbol{\theta} \mid \boldsymbol{y})]^2 \pi(\boldsymbol{y}, \boldsymbol{\theta} \mid \xi) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y}$$
$$= -\int \mathrm{tr} \left\{ \mathrm{Var}(\boldsymbol{\theta} \mid \boldsymbol{y}, \xi) \pi(\boldsymbol{y} \mid \xi) \right\} \, \mathrm{d}\boldsymbol{y}$$

- the **expected negative squared error loss (NSEL)**

Challenges

In general, Bayesian design is easy in principle but hard in practice

1. For most nonlinear models, the **expected utility will be intractable** and involves high-dimensional integrals with respect to $\boldsymbol{y}$
   - often, obtaining the utility function itself requires the solution of intractable integrals (cf both ESIG and NSEL)
   - numerical or analytical approximation is required (eg Ryan et al. 2016)
2. A **high-dimensional optimisation problem** results for multi-factor experiments with many design points

Asymptotic approximations

For large n, the inverse information matrix $M(\xi;\, \boldsymbol{\theta})$ is an asymptotic approximation to the posterior variance-covariance matrix

Using this approximation, we can define Bayesian analogues of classical optimality criteria

D-optimality: maximise

$$U_D(\xi) = \int \log \det M(\xi;\, \boldsymbol{\theta}) \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$$

- approximation to ESIG

A-optimality: maximise

$$U_A(\xi) = -\int \mathrm{tr} M^{-1}(\xi;\, \boldsymbol{\theta}) \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$$

- approximation to NSEL

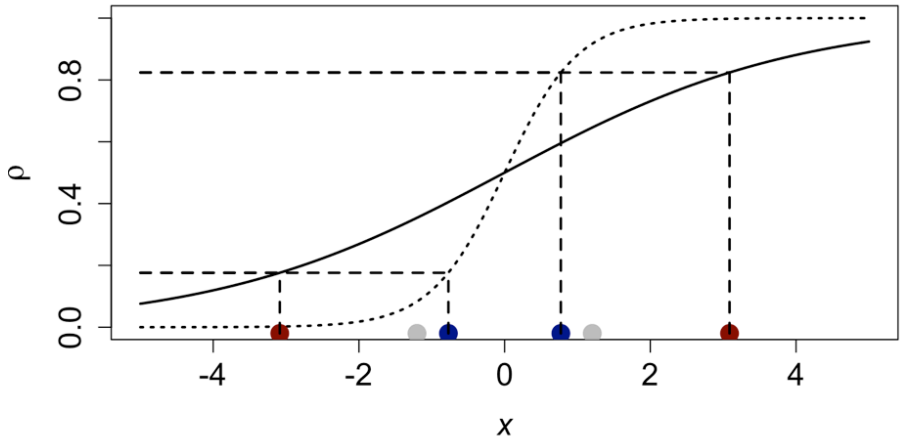
These integrals, with respect to $\boldsymbol{\theta}$, are lower dimensional and more amenable to deterministic (quadrature) approximation, eg Gotwalt, Jones, and Steinberg (2009)

The acebayes package provides functions for constructing approximations to expected utilities

- default is to use quadrature to approximate the Bayesian D-optimality objective function

```
library(acebayes)
prior <- list(support = matrix(c(0, 0, .5, 2), nrow = 2))
logreg.util <- utilityglm(formula = ~ x, family = binomial, prior = prior)$utility
BDcrit <- function(xi) logreg.util(data.frame(x = xi))
bdopt <- optim(par = c(-1, 1), BDcrit, control = list(fnscale = -1))
bdopt$par
```

```
## [1] -1.202960  1.203132
```



Monte Carlo approximation

As an alternative to analytical approximations, Monte Carlo approximation to the expected utility is simple to implement and intuitively appealing

$$\tilde{U}(\xi) = \frac{1}{B} \sum_{i=1}^{B} \tilde{u}(\xi, \boldsymbol{y}_i, \boldsymbol{\theta}_i)$$

where

- $\left\{\boldsymbol{\theta}_h, \boldsymbol{y}_h\right\}_{h=1}^{B}$ is a random sample from $\pi(\boldsymbol{\theta}, \boldsymbol{y} \mid \xi)$
- $\tilde{u}(\xi, \boldsymbol{y}, \boldsymbol{\theta})$ is, where necessary, an approximation to the utility function (often, nested Monte Carlo is required)

How to construct the approximation $\tilde{u}(\xi, \boldsymbol{y}, \boldsymbol{\theta})$ is an active area of research, eg Overstall, McGree, and Drovandi (2018), Beck et al. (2018)

Optimisation

Find an optimal design using Monte Carlo:

```
priorMC <- function(B) cbind(rep(0, B), runif(n = B, min = .5, max = 2))
logreg.utilSIG <- utilityglm(formula = ~ x, family = binomial, prior = priorMC, criterion = "SIG")$utility
BDcritSIG <- function(xi, B = 1000) mean(logreg.utilSIG(data.frame(x = xi), B))
bdoptSIG <- optim(par = c(-1, 1), BDcritSIG, control = list(fnscale = -1))
bdoptSIG$par
```

```
## [1] -1.205382  1.182047
```

```
bdopt$par
```

```
## [1] -1.202960  1.203132
```

Larger Monte Carlo sample sizes will produce results more similar to the design found using quadrature (in this example)

In general, direct optimisation of the Monte Carlo approximation requires large B to generate suitable smooth objective function and/or expensive stochastic algorithms (eg genetic algorithms)

Hamada et al. (2001)

Alternatively, the optimisation can be embedded within a simulation scheme and samples generated from the joint artificial distribution of $\xi, \boldsymbol{y}, \boldsymbol{\theta}$

- take ξ^*, the optimal design, to be the posterior mode of the marginal distribution
- most effective for small experiments (both numbers of variables and runs)

Müller (1999), Müller, Sansó, and De Iorio (2004)

Smoothing-based optimisation

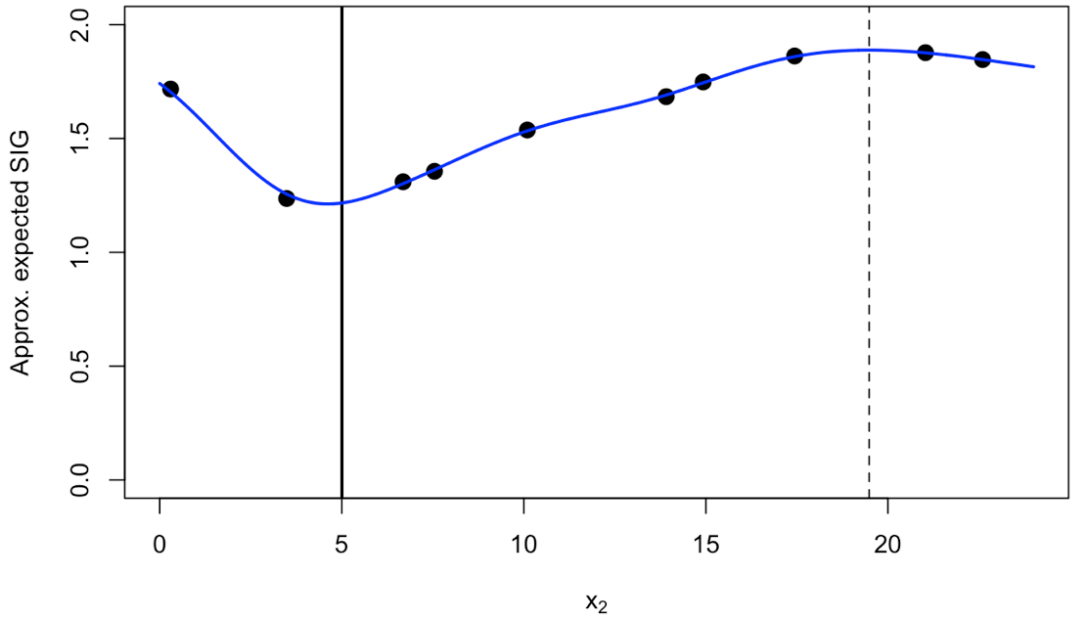
Instead of directly minimising a Monte Carlo approximation to the expected utility, find designs via **curve fitting** (Müller and Parmigiani 1996)

1. Evaluate the Monte Carlo approximation $\tilde{U}(\xi)$ for a small number of designs, $\xi_1, \ldots, \xi_Q$
2. Smooth the "data" $\left\{\xi_i, \tilde{U}(\xi_i)\right\}$, i.e. fit a statistical model, to obtain a surrogate $\hat{U}(\xi)$
3. Find ξ that maximises $\hat{U}(\xi)$

Return to Example 1, compartmental model

- find a design with $n = 2$ runs, with **fixed** $x_1 = 5$
- use Monte Carlo approximation to SIG for 10 values of x_2

```
library(DiceKriging)
library(DiceDesign)
n <- 10; x1<- -0.583; x2 <- 2 * maximinSA_LHS(lhsDesign(n, 1)$design)$design- 1
u <- NULL; for(i in 1:n) u[i] <- mean(utilcomp15sig(c(x1, x2[i]), B = 1000))
par(mar = c(4, 4, 2, 2) + 0.1)
plot(12 * (x2 + 1), u, xlab = expression(x[2]), ylab = "Approx. expected SIG", xlim = c(0, 24),
     ylim = c(0, 2), pch = 16, cex = 1.5); abline(v = 12 * (x1 + 1), lwd = 2)
usmooth <- km(design = 12 * (x2 + 1), response = u, nugget = 1e-3, control = list(trace = F))
xgrid <- matrix(seq(0, 24, l = 1000), ncol = 1); pred <- predict(usmooth, xgrid, type = "SK")$mean
lines(seq(0, 24, l = 1000), pred, col = "blue", lwd = 2); abline(v = xgrid[which.max(pred), ], lty = 2)
```



Approximate coordinate exchange

Coordinate exchange, a version of cyclic ascent, is a popular algorithm for finding optimal designs (Meyer and Nachtsheim 1995)

- optimisation of $\xi = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ proceeds coordinate-wise, i.e. just one of the x_{ij} is varied at a time

Approximate coordinate exchange (ACE) combines coordinate exchange with smoothing to find high-dimensional designs under computationally expensive approximate expected utilities

- a nonparametric regression model (a Gaussian process) is used to smooth the Monte Carlo approximations of $U(\xi)$ as a function of one coordinate
  - reduces the computational burden
  - facilitates optimisation of a noisy function

Overstall and Woods (2017)

Return to the multifactor logistic regression (crystallography) example

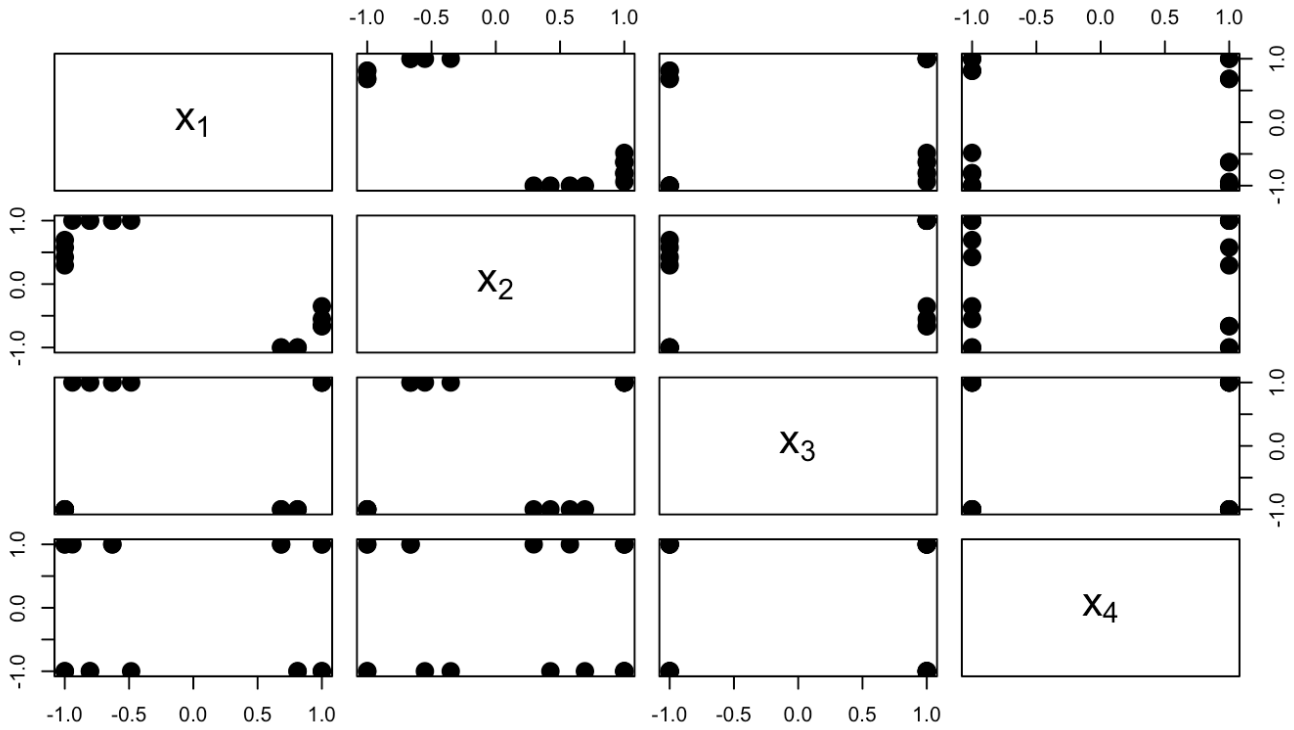
```
## set up prior
priorMFL <- function(B) {
  b0 <- runif(B, -3, 3)
  b1 <- runif(B, 4, 10)
  b2 <- runif(B, 5, 11)
  b3 <- runif(B, -6, 0)
  b4 <- runif(B, -2.5, 2.5)
  cbind(b0, b1, b2, b3, b4)
}
## define the utility function
MFL.utilSIG <- utilityglm(formula = ~ x1 + x2 + x3 + x4, family = binomial, prior = priorMFL,
                          criterion = "SIG")$utility
## starting design with n=18 runs, on [-1, 1]
d <- 2 * randomLHS(18, 4) - 1
colnames(d) <- paste0("x", 1:4)
## approximate expected utility for starting design
mean(MFL.utilSIG(d, 1000))
```

```
## [1] 1.139546
```

```
## not run - quite computationally expensive
MLF.ace <- ace(utility = MFL.utilSIG, start.d = d, progress = T)
```

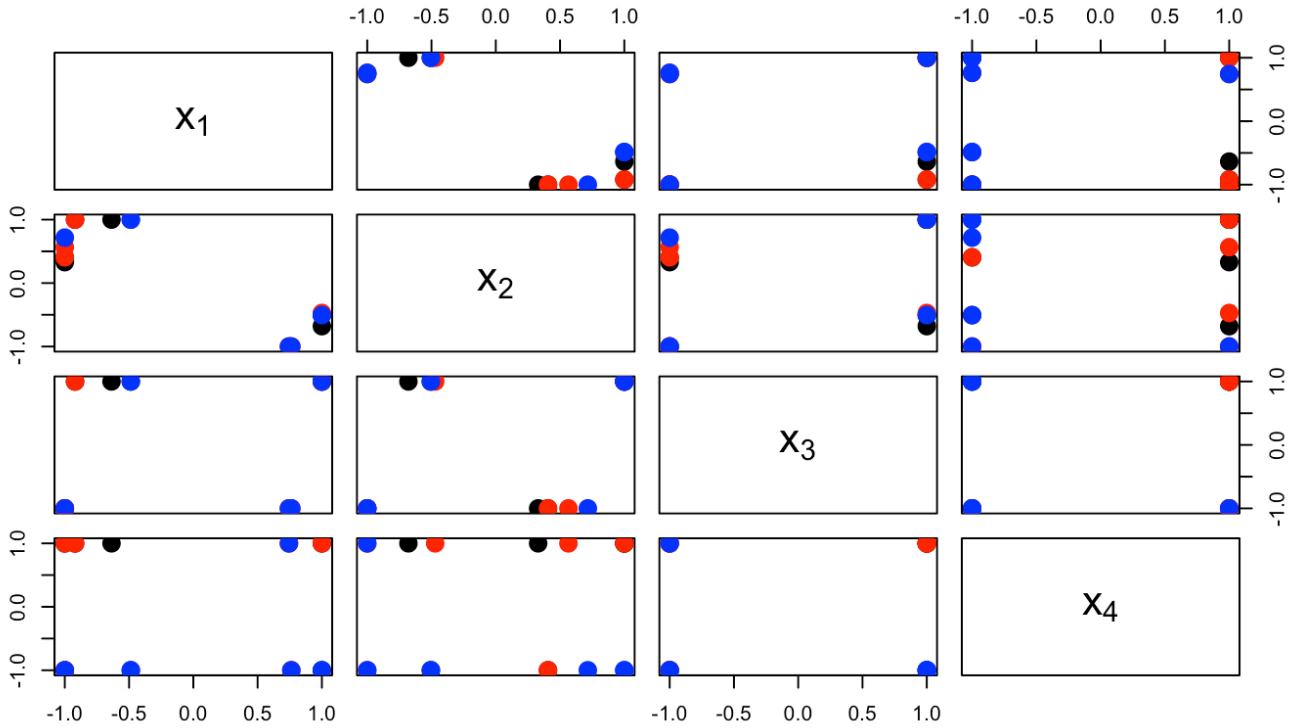
For **this** logistic regression example, acebayes has some designs precomputed

```
pairs(optdeslrsig(18), pch = 16,
      labels=c(expression(x[1]), expression(x[2]), expression(x[3]), expression(x[4])), cex = 2)
```



Hierachical logistic regression with $g = 3$ groups (blocks, eg wellplates)

```
pairs(optdeshlrsig(18), pch = 16,
      labels=c(expression(x[1]), expression(x[2]), expression(x[3]), expression(x[4])),
      col = c("black", "red", "blue")[rep(1:3, rep(6, 3))], cex = 2)
```

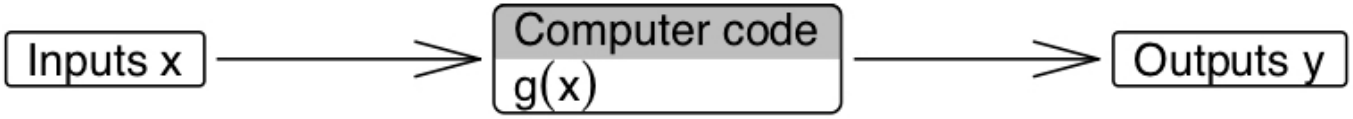


Computer experiments

Introduction

Many physical and social processes can be approximated by computer codes which encapsulate mathematical models

- eg partial differential equations solved using finite element methods
- eg reaction kinetics modelling in computational biology, in-silico chemistry



- computer code: numerical implementation of the mathematical model

Key feature: the model does not have a closed-form; it can only be evaluated numerically, and this is typically (relatively) expensive

We will focus on **deterministic** computer models

Computer experiments

Assumption: $g(\boldsymbol{x})$ can only be evaluated numerically; i.e. $g(\boldsymbol{x})$ can be computed for a given $\boldsymbol{x}$ but the general form is unknown

How do we learn about the function $g(\boldsymbol{x})$?

In an analogy to a physical system, we experiment on $g(\boldsymbol{x})$, i.e.

- choose a design $\xi = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$
- evaluate $g(\boldsymbol{x}_i)$ (run the computer code)

Use the "data" $\{\boldsymbol{x}_i, g(\boldsymbol{x}_i)\}$ to build statistical models linking $\boldsymbol{x}$ and $g(\boldsymbol{x})$

- called **emulators**; typically use a Gaussian process

See Santner, Williams, and Notz (2003)

(Very) simple example

Climate modelling involves the solution of many intractable equations, leading to mathematical models evaluated via computationally expensive computer codes

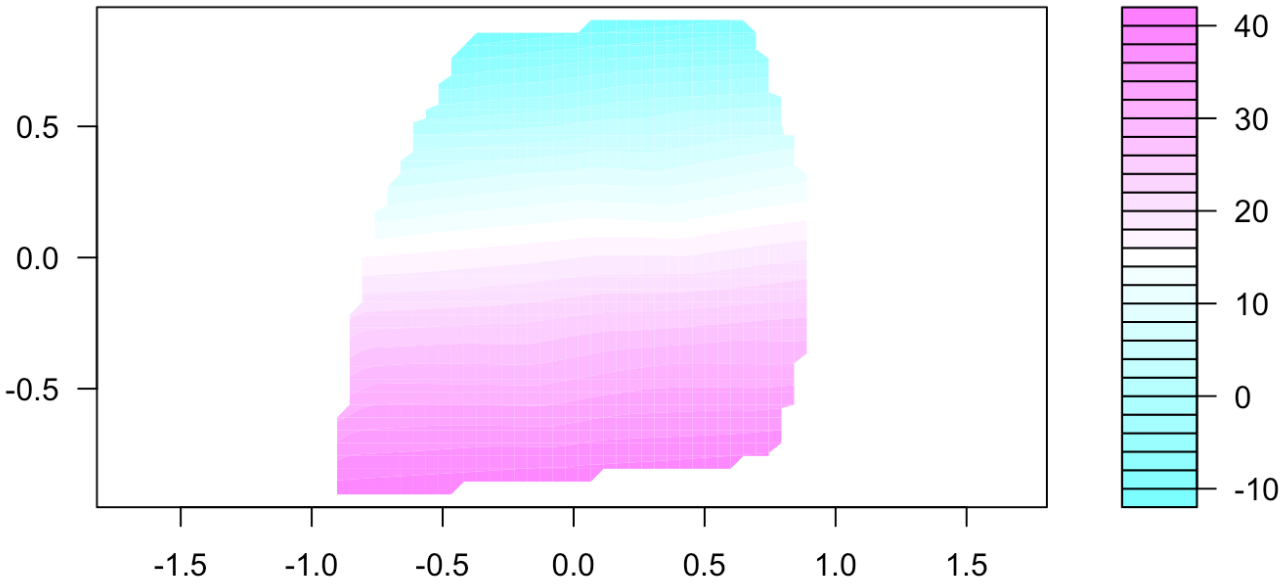
- lots of applications of computer experiments

We will illustrate methods on a very simple example: a time-stepping advective/diffusive surface layer meridional EBM (energy balance model)

- 2D earth with no land
- each surface object has a percentage of ice cover
- different albedo (fraction of solar energy reflected) for ice vs non-ice surfaces
- ocean circulation is explicitly modelling (cf Atlantic gulf stream)
- two variables: x_1 - solar constant; x_2 - non-ice albedo
- output is mean temperature

See https://wiki.aston.ac.uk/foswiki/bin/view/MUCM/SurfebmModel

```
## design and data are in 'ebm'
library(akima)
fld <- interp(x = ebm$x1, y = ebm$x2, z = ebm$y)
filled.contour(x = fld$x, y = fld$y, z = fld$z, asp = 1)
```



Space-filling designs

As we will see later, emulators are usually constructed using nonparametric statistical models

This choice leads naturally to using **space-filling** designs

- such designs do not rely on the functional form of the relationship between the code inputs and the response
- good coverage is important for prediction (we will predict "better" near points we have already run the computer model)

Common designs are chosen to optimise some space-filling metric, or formed from (stratified) random sampling

Space-filling designs do not have **replication**, so ideal for deterministic computer models

Uniform designs

Many designs proposed for computer experiments are related to ideas underpinning quadrature, and the approximation of an expectation.

Let $\bar{g} = \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{x}_i)$, the sample mean of $g(\cdot)$ for ξ. Then

$$|E_{\boldsymbol{x}}[g(\boldsymbol{x})] - \bar{g}| \leq \text{constant} \times D(\xi)$$

where $D(\xi)$ is the **star discrepancy** of the design

- $D(\xi)$ is a measure of the uniformity of the design points

This relationship leads to the criterion of design selection via minimising discrepancy

- $D(\xi)$ is difficult to compute for moderate to high numbers of dimensions
- therefore, it is more common to minimise the related **centred L_2-discrepancy**

Fang, Li, and Sudjianto (2006), Ch.3

Designs based on measures of distance

Two sensible criteria for the selection of a space-filling design are

- make sure no two points in the design are too close together
- make sure no point in the design region is too far from a design point

(Johnson, Moore, and Ylvisaker 1990)

The Euclidean distance between points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is given by

$$\delta(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\sum_{i=1}^{k} \left(x_j - x_j'\right)^2}$$

Mm and mM designs

Using Euclidean distance, we can define

- maximin (Mm) criterion: maximise

$$\min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \xi} \delta(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- minimax (mM) criterion: minimise

$$\max_{\boldsymbol{x}} \delta(\boldsymbol{x}, \xi)$$

where the distance between a point $\boldsymbol{x}$ and a design ξ is defined as

$$\delta(\boldsymbol{x}, \xi) = \min_{\boldsymbol{x}_j \in \xi} \delta(\boldsymbol{x}, \boldsymbol{x}_i)$$

Roughly speaking, an Mm design **spreads** out the design points, and an mM design **covers** the design region

Intuitively, covering the design region seems more desirable (eg for prediction), but optimising the mM objective function is computationally challenging. Hence, Mm designs are more commonly used

Latin hypercube designs

For high-dimensional problems, space-filling is difficult

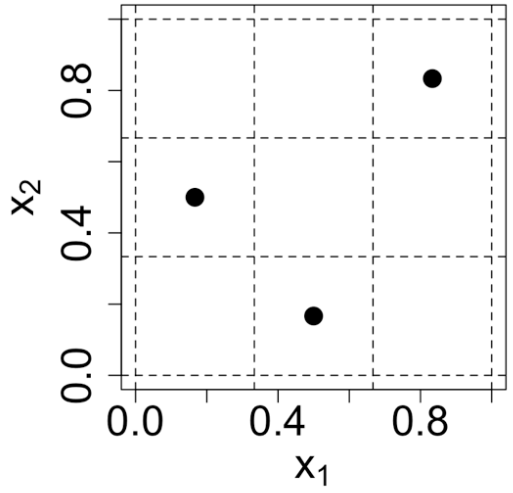
- many points are required to adequate space-fill a high-dimensional space (curse of dimensionality)

Latin hypercube designs (LHDs) are randomly chosen sets of points with the restriction of uniform one-dimensional projections (McKay, Beckman, and Conover 1979)

- each variable has no overlapping points, and good coverage (compare with a factorial design, which has hidden replication)
- can be easily constructed using permutations of integers

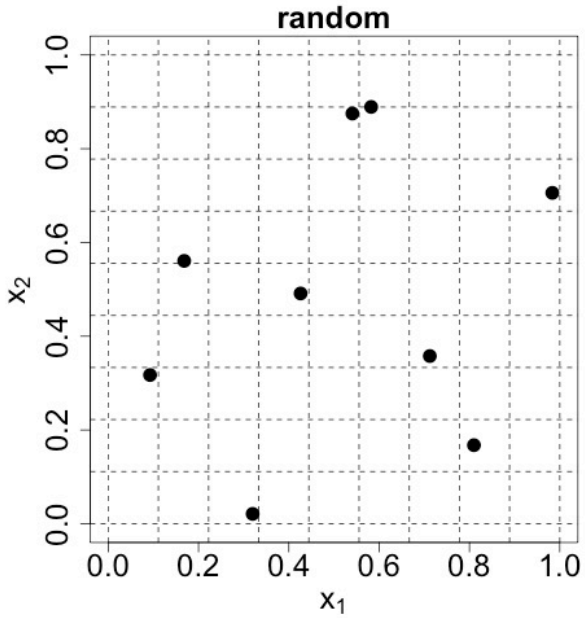
An LHD only guarantees space-filling properties in each one-dimensional projection, not overall. So we normally combine the Latin hypercube principle with a space-filling criteria, eg to find a Mm LHD

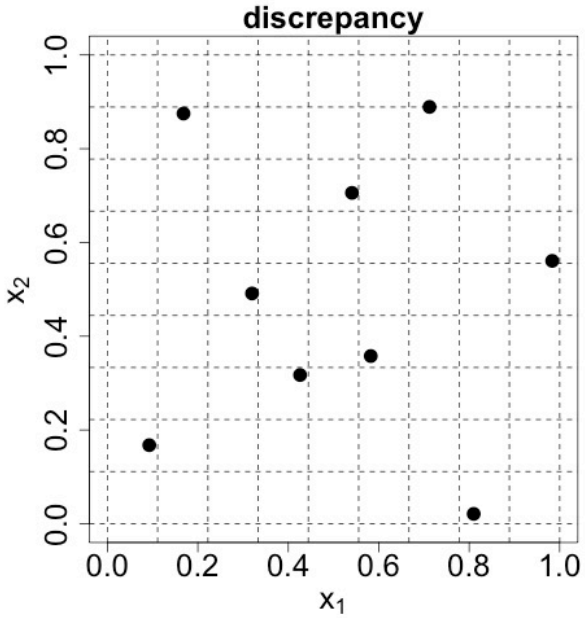
```r
LH <- function(n = 3, d = 2) {
    D <- NULL
    for(i in 1:d) D <- cbind(D, sample(1:n, n))
    D
}
set.seed(4)
par(mar=c(5,6,2,4)+0.1, pty = "s")
plot((LH() -.5)/ 3, xlim = c(0, 1), ylim = c(0, 1), pty = "s", xlab = expression(x[1]),
     ylab = expression(x[2]), pch = 16, cex.lab = 2, cex.axis = 2, cex = 2)
abline(v = c(0, 1/3, 2/3, 1), lty = 2)
abline(h = c(0, 1/3, 2/3, 1), lty = 2)
```

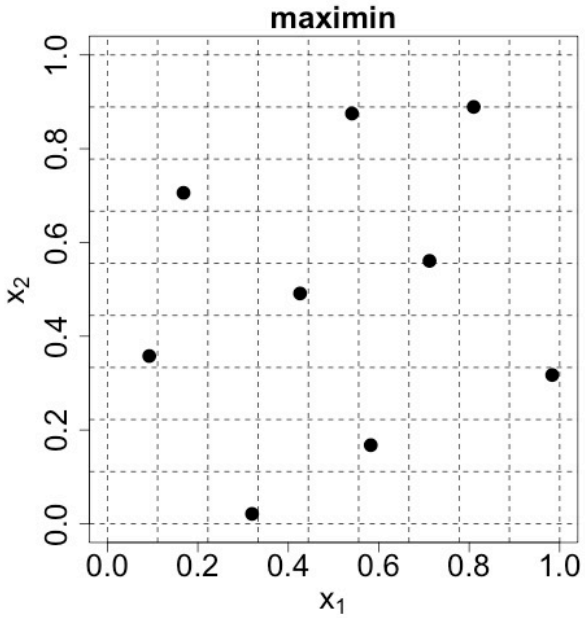


The DiceDesign package has functions to generate various LHDs

```
library(DiceDesign)
lhs.d <- lhsDesign(9, 2)
plot(lhs.d$design, xlim = c(0, 1), ylim = c(0, 1), pty = "s", xlab = expression(x[1]),
     ylab = expression(x[2]), pch = 16, cex.lab = 2, cex.axis = 2, cex = 2,
     main = "random", cex.main = 2)
abline(v = seq(0, 9) / 9, lty = 2)
abline(h = seq(0, 9) / 9, lty = 2)

discrep.d <- discrepSA_LHS(lhs.d$design, criterion = "C2")
plot(discrep.d$design, xlim = c(0, 1), ylim = c(0, 1), pty = "s", xlab = expression(x[1]),
     ylab = expression(x[2]), pch = 16, cex.lab = 2, cex.axis = 2, cex = 2,
     main = "discrepancy", cex.main = 2)
abline(v = seq(0, 9) / 9, lty = 2)
abline(h = seq(0, 9) / 9, lty = 2)

maximin.d <- maximinSA_LHS(discrep.d$design)
plot(maximin.d$design, xlim = c(0, 1), ylim = c(0, 1), pty = "s", xlab = expression(x[1]),
     ylab = expression(x[2]), pch = 16, cex.lab = 2, cex.axis = 2, cex = 2,
     main = "maximin", cex.main = 2)
abline(v = seq(0, 9) / 9, lty = 2)
abline(h = seq(0, 9) / 9, lty = 2)
```

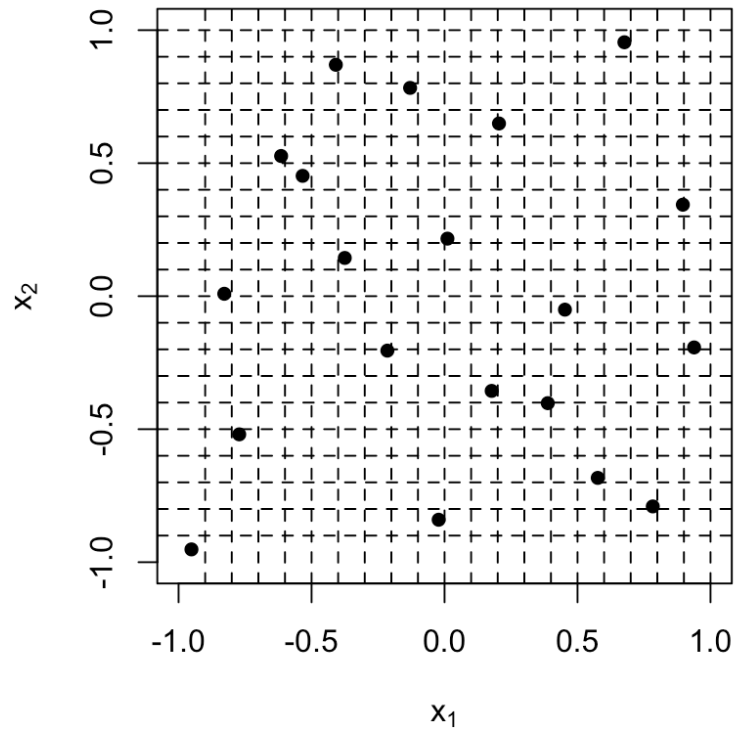

random
x2
x1
0.0
0.2
0.4
0.6
0.8
1.0


discrepancy
x2
x1
0.0
0.2
0.4
0.6
0.8
1.0


maximin
x2
x1
0.0
0.2
0.4
0.6
0.8
1.0

The design for the EBM example is a Mm LHD

```
par(mar=c(5,6,2,4)+0.1, pty = "s")
plot(ebm[, 2:3], xlim = c(-1, 1), ylim = c(-1, 1), pty = "s", xlab = expression(x[1]),
     ylab = expression(x[2]), pch = 16, asp = 1)
abline(v = 2 * seq(0:20) / 20 - 1, lty = 2)
abline(h = 2 * seq(0:20) / 20 - 1, lty = 2)
```



Gaussian process

The most common statistical model used to emulate computer models is the Gaussian process (GP)

- flexible, nonparametric regression model (few assumptions made about $g(\boldsymbol{x})$)
- naturally allows for uncertainty quantification (eg prediction intervals)
- interpolates observed responses

An intuitive way to think about a GP is as a prior for the unknown function $g(\boldsymbol{x})$ within a Bayesian framework

We say that

$$g(\boldsymbol{x}) \sim \mathrm{GP}\left(\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta}, \sigma^2\kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})\right) ,$$

where $\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta}$ is the mean, $\kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\phi})$ is the correlation function, $\boldsymbol{\theta}$ is the vector of correlation parameters and σ^2 is the constant variance, if:

- any vector $\boldsymbol{g} = (g(\boldsymbol{x}_1), \ldots, g(\boldsymbol{x}_n))^{\mathrm{T}}$ satisfies

$$\boldsymbol{g} \sim N\left(F\boldsymbol{\beta}, \sigma^2 K(\boldsymbol{\theta})\right) ,$$

  with F a model matrix and K the $m \times m$ covariance matrix defined by $K(\boldsymbol{\theta})_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})$.

See Rasmussen and Williams (2006)

Typically, very simple mean functions are chosen for the GP, eg

- constant: $\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta} = \beta_0$ (sometimes called ordinary kriging)
- linear: $\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta} = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$ (universal kriging)

The most commonly used correlation functions are **separable** and **stationary**

- squared exponential:

$$\kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \exp\left[- \sum_j \left(\frac{|x_j - x_j'|}{\theta_j} \right)^2 \right]$$

- Matérn $\nu = 5/2$

$$\kappa(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \prod_j \left(1 + \sqrt{5} \frac{|x_j - x_j'|}{\theta_j} + \frac{5}{3} \left(\frac{|x_j - x_j'|}{\theta_j} \right)^2 \right) \exp\left(-\sqrt{5} \frac{|x_j - x_j'|}{\theta_j} \right)$$

The Matérn function can be defined for other values of ν; for $\nu \to \infty$, the squared exponential function is obtained

Given model evaluations $\boldsymbol{g} = [g(\boldsymbol{x}_1), \dots, g(\boldsymbol{x}_n)]$, a posterior GP can be obtained:

$$g(\boldsymbol{x}) \mid \boldsymbol{g}, \boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2 \sim N\left(m(\boldsymbol{x}), s^2(\boldsymbol{x})\right)$$

- $m(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\beta} + \boldsymbol{\kappa}_n^{\mathrm{T}} K^{-1}(\boldsymbol{g} - F\boldsymbol{\beta})$
- $s^2(\boldsymbol{x}) = \sigma^2\left(1 - \boldsymbol{\kappa}_n^{\mathrm{T}} K^{-1}\boldsymbol{\kappa}_n\right)$

where $\boldsymbol{\kappa}_n = [\kappa(\boldsymbol{x}, \boldsymbol{x}_i\,;\,\boldsymbol{\theta})]_{i=1}^n$ is a vector of correlations between $g(\boldsymbol{x})$ and $g(\boldsymbol{x}_1), \dots, g(\boldsymbol{x}_n)$

The updating of the prior mean and variance depends on the "distance" between $\boldsymbol{x}$ and the points in ξ

- the posterior mean will be adjusted more for points closer to the design
- predictions at these points will have smaller posterior variance

If $\boldsymbol{x} = \boldsymbol{x}_i$ (so we are predicting at a design point), $K^{-1}\boldsymbol{\kappa}_n = \boldsymbol{e}_i$, the ith unit vector

- $m(\boldsymbol{x}_i) = \boldsymbol{f}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\beta} + \boldsymbol{e}_i^{\mathrm{T}}(\boldsymbol{g} - F\boldsymbol{\beta}) = g(\boldsymbol{x}_i)$
- $s^2(\boldsymbol{x}_i) = \sigma^2\left(1 - \boldsymbol{\kappa}_n^{\mathrm{T}}\boldsymbol{e}_i\right) = \sigma^2\left(1 - \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i\,;\,\boldsymbol{\theta})\right) = 0$

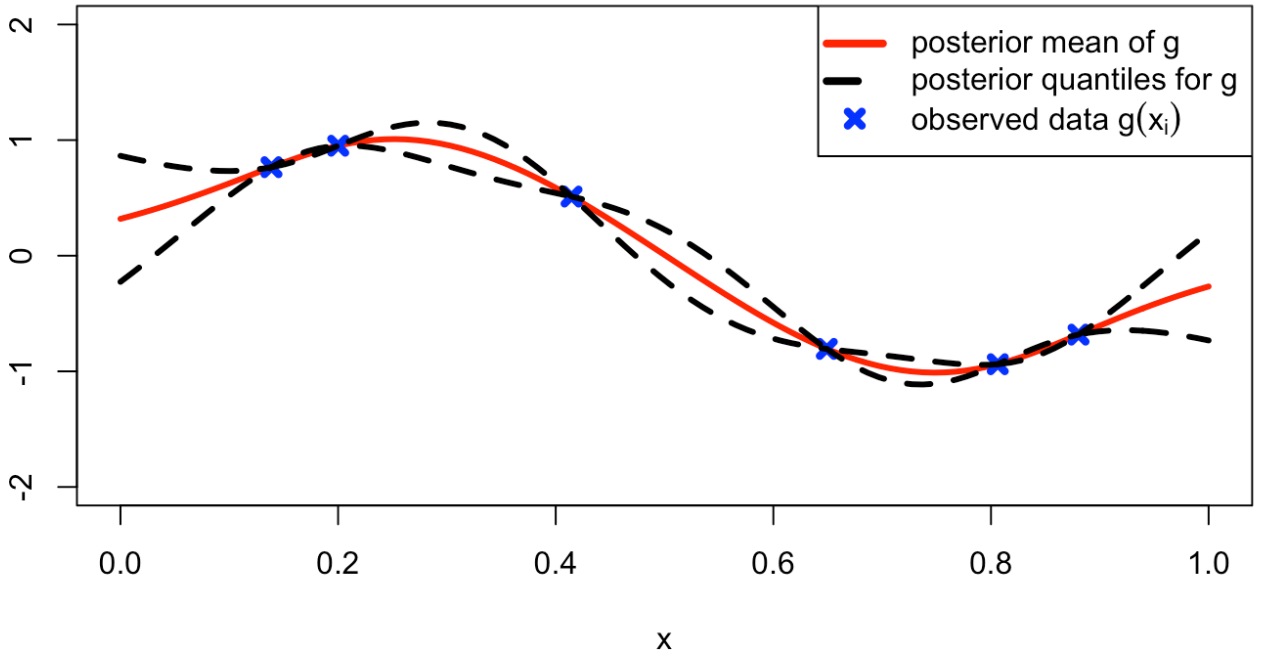
The posterior GP **interpolates** - exactly what you want for a deterministic computer code

Inference unconditional on all the hyperparameters requires numerical approximation (eg Markov chain Monte Carlo)

- it is common to estimate the parameters, eg using maximum likelihood, to "plug-in" to the posterior predictive distribution

A simple example: $g(x) = \sin(2\pi x)$ using the DiceKriging package

```
library(DiceDesign)
library(DiceKriging)
xi <- lhsDesign(6, 1)$design
y <- sin(2 * pi * xi)
gp <- km(design = xi, response = y, control = list(trace = F))
xs <- sort(c(seq(0, 1, length = 100), xi))
gpp <- predict(gp, newdata = xs, type = "SK")

plot(xs, gpp$mean, ylim = c(-2, 2), type = "l", col = "red", lwd = 3, ylab = "", xlab = "x")
points(xi, y, pch = 4,lwd = 4, col = "blue")
lines(xs, gpp$upper95, lty = 2, lwd = 3)
lines(xs, gpp$lower95, lty = 2, lwd = 3)
legend(x = "topright", legend = c("posterior mean of g", "posterior quantiles for g",
                                  expression(paste("observed data ", g(x[i])))), lty = c(1, 2, NA),
       pch = c(NA, NA, 4), lwd = c(4, 4, 4), col = c("red", "black", "blue"))
```

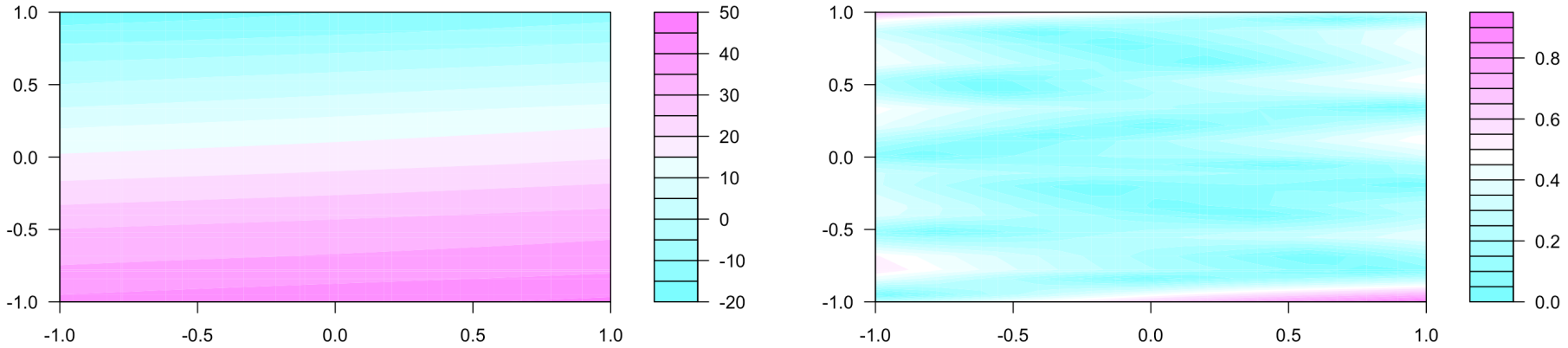

posterior mean of g
posterior quantiles for g
observed data g(x_i)
2
1
0
-1
-2
0.0
0.2
0.4
0.6
0.8
1.0
x

Return to the EBM example

```
gpebm <- km(formula = ~., design = ebm[, 2:3], response = ebm[, 1], control = list(trace = F))
gpebm
```

```
##
## Call:
## km(formula = ~., design = ebm[, 2:3], response = ebm[, 1], control = list(trace = F))
##
## Trend  coeff.:
##                Estimate
##  (Intercept)    16.3262
##           x1     2.4078
##           x2   -28.9973
##
## Covar. type  : matern5_2
## Covar. coeff.:
##                Estimate
##    theta(x1)     2.8829
##    theta(x2)     0.2722
##
## Variance estimate: 2.215045
```

```
xs1 <- sort(c(seq(-1, 1, length = 10), ebm[, 2]))
xs2 <- sort(c(seq(-1, 1, length = 10), ebm[, 3]))
xs <- expand.grid(x1 = xs1, x2 = xs2)
gppebm <- predict(gpebm, newdata = xs, type = "UK")
filled.contour(x = xs1, y = xs2, z = matrix(gppebm$mean, nrow = length(xs1)))
filled.contour(x = xs1, y = xs2, z = matrix(gppebm$sd, nrow = length(xs1)))
```



Bayesian optimisation

A common task is **optimisation** of $g(\boldsymbol{x})$

When $g(\boldsymbol{x})$ is computationally expensive to evaluate, computer experiments and emulators can be used to facilitate the optimisation.

The field of **Bayesian optimisation** uses sequentially collected evaluations of $g(\boldsymbol{x})$

- place a prior distribution (eg GP) on $g(\boldsymbol{x})$
- collect function evaluations at points chosen sequentially via an acquisition function
- update the prior to a posterior distribution, and infer the maximum/minimum of $g(\boldsymbol{x})$

Uncertainty in the posterior (i.e. for $g(\boldsymbol{x})$ at unobserved $\boldsymbol{x}$) leads to **exploration/exploitation** trade-off

The most common acquisition function is **expected improvement (EI)**

See Jones, Schonlau, and Welch (1998)

For a deterministic computer model and a minimisation problem, the improvement from performing one more run is given by:

$$\max(g_{\min} - g(\boldsymbol{x}), 0)$$

where $g_{\min}$ is the minimum across the model runs performed to date

This quantity is a random variable - we are uncertain about $g(\boldsymbol{x})$ at a point we have not observed.

EI chooses $\boldsymbol{x}$ to maximise

$$E_g\left[\max(g_{\min} - g(\boldsymbol{x}), 0)\,;\,\boldsymbol{g}\right] = \left[g_{\min} - m(\boldsymbol{x})\right]\Phi\left(\frac{g_{\min} - m(\boldsymbol{x})}{s(\boldsymbol{x})}\right) + s(\boldsymbol{x})\phi\left(\frac{g_{\min} - m(\boldsymbol{x})}{s(\boldsymbol{x})}\right)$$

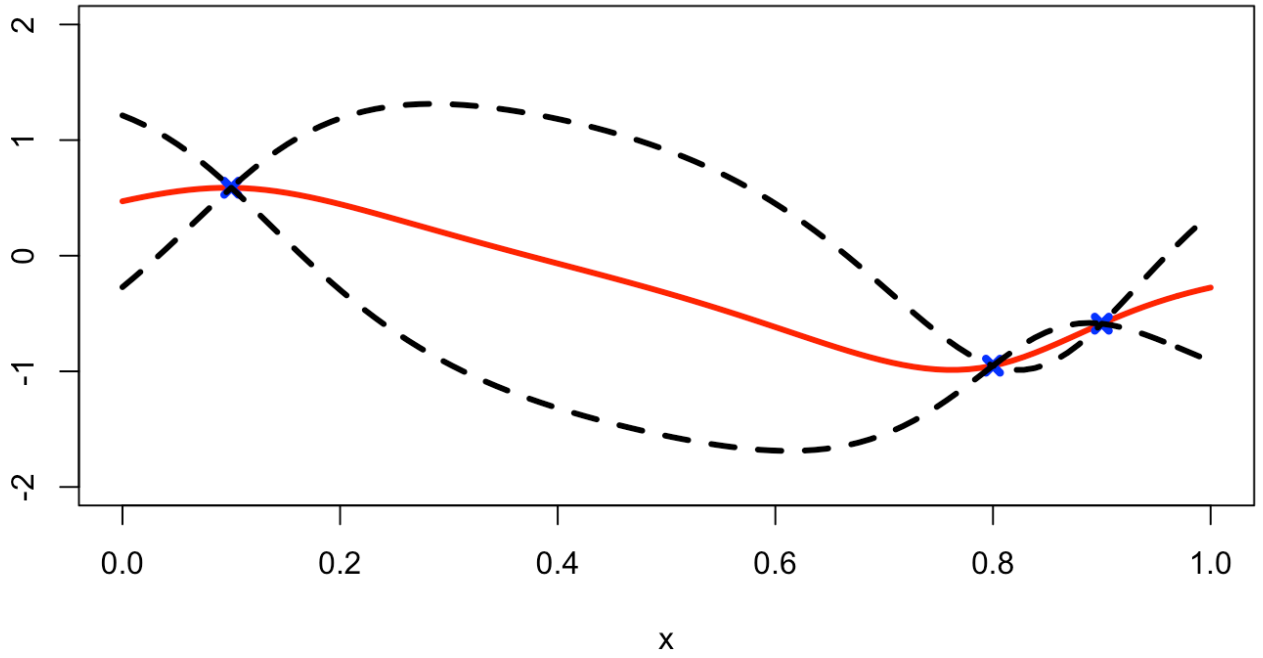
where ϕ and Φ are the standard normal pdf and cdf, respectively

EI is an decreasing function of $m(\boldsymbol{x})$ and an increasing function of $s^2(\boldsymbol{x})$, so it leads to choosing design points that either minimise the posterior mean or the posterior variance

- experiment either where our uncertainty is high or near where we predict the minimum to be (**explore or exploit**)

A simple example: $g(x) = \sin(2\pi x)$ but with a different starting design using DiceOptim

```
xi <- matrix(c(0.1, 0.8, 0.9), ncol = 1)
fn <- function(x) sin(2 * pi * x)
y <- fn(xi)
gp <- km(design = xi, response = y, control = list(trace = F))
xs <- sort(c(seq(0, 1, length = 100), xi))
gpp <- predict(gp, newdata = xs, type = "SK")

plot(xs, gpp$mean, ylim = c(-2, 2), type = "l", col = "red", lwd = 3, ylab = "", xlab = "x")
points(xi, y, pch = 4,lwd = 4, col = "blue")
lines(xs, gpp$upper95, lty = 2, lwd = 3)
lines(xs, gpp$lower95, lty = 2, lwd = 3)
```


2
1
0
-1
-2
0.0
0.2
0.4
0.6
0.8
1.0
x

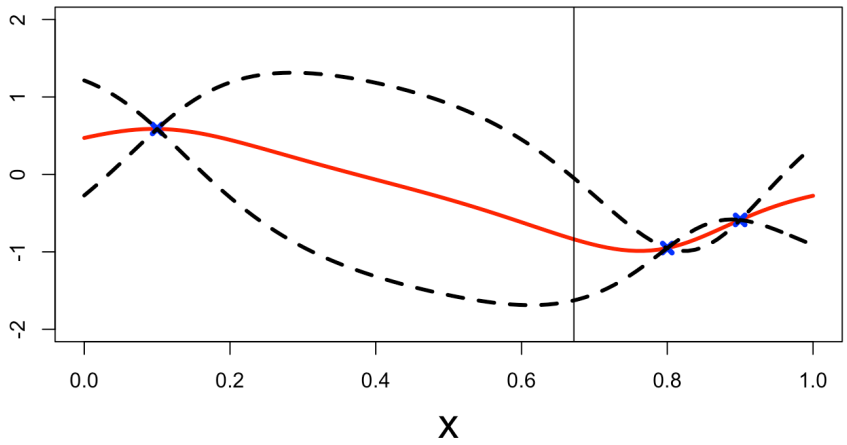
```
library(DiceOptim)
xin <- max_EI(model = gp, lower = 0, upper = 1)$par
```

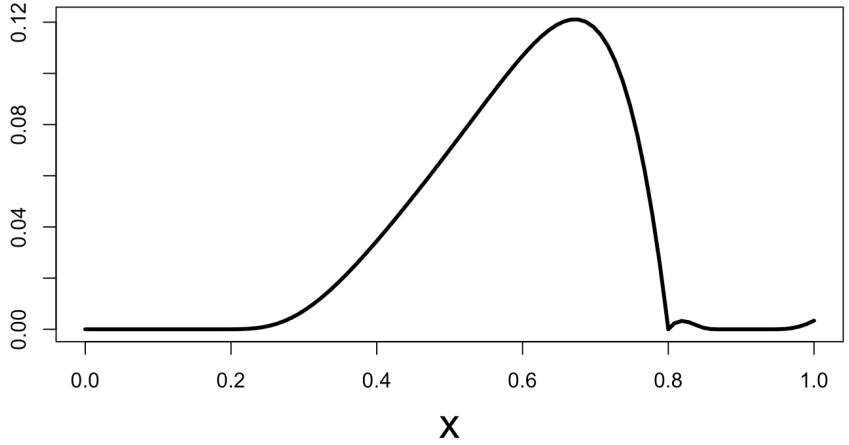
```
##
##
## Mon Aug 26 15:55:26 2019
## Domains:
##  0.000000e+00   <=  X1   <=    1.000000e+00
##
## NOTE: The total number of operators greater than population size
## NOTE: I'm increasing the population size to 10 (operators+1).
##
## Data Type: Floating Point
## Operators (code number, name, population)
##  (1) Cloning.........................  0
##  (2) Uniform Mutation................  1
##  (3) Boundary Mutation...............  1
##  (4) Non-Uniform Mutation............  1
##  (5) Polytope Crossover..............  1
##  (6) Simple Crossover................  2
##  (7) Whole Non-Uniform Mutation.....  1
##  (8) Heuristic Crossover............  2
##  (9) Local-Minimum Crossover.........  0
##
## HARD Maximum Number of Generations: 12
## Maximum Nonchanging Generations: 2
## Population size       : 10
## Convergence Tolerance: 1.000000e-21
##
## Using the BFGS Derivative Based Optimizer on the Best Individual Each Generation.
```

```
EI(xin, gp)
```

```
## [1] 0.1211283
```

```
plot(xs, gpp$mean, ylim = c(-2, 2), type = "l", col = "red", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
points(xi, y, pch = 4,lwd = 4, col = "blue")
lines(xs, gpp$upper95, lty = 2, lwd = 3)
lines(xs, gpp$lower95, lty = 2, lwd = 3)
abline(v = xin)
plot(xs, sapply(xs, EI, model = gp), type = "l", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
```



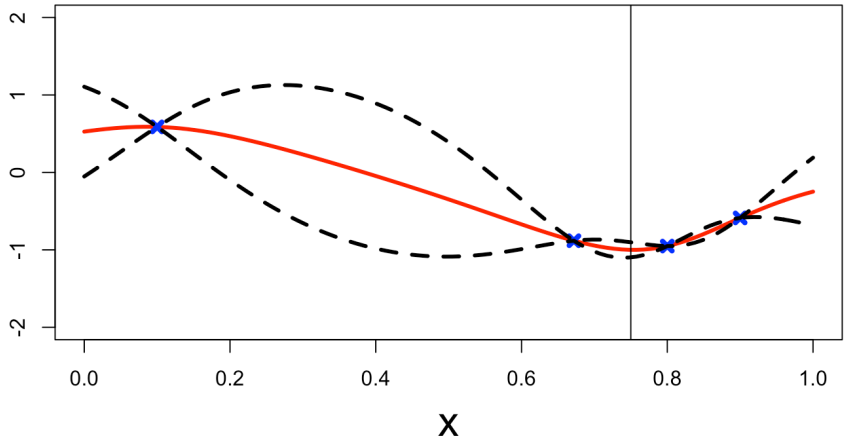


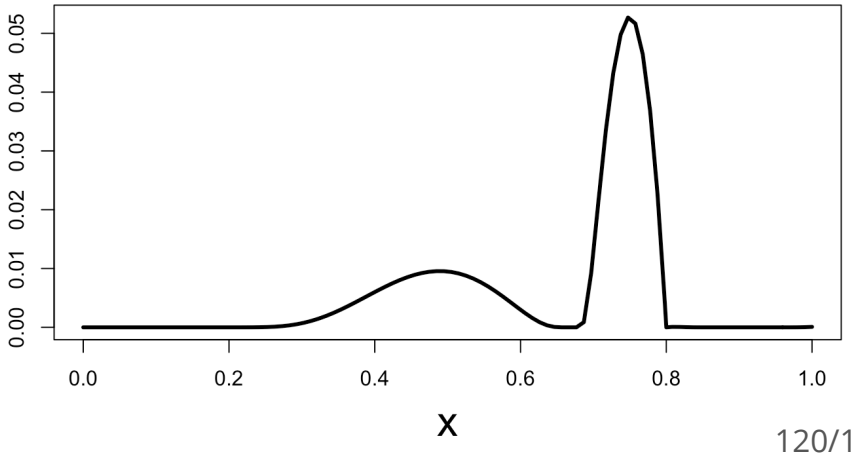
```
xi <- rbind(xi, xin)
y <- c(y, fn(xin))
gp2 <- km(design = xi, response = y, control = list(trace = F))
xin <- max_EI(model = gp2, lower = 0, upper = 1, control = list(trace = F))$par


##
##
## Mon Aug 26 15:55:26 2019
## Domains:
##  0.000000e+00   <=  X1   <=    1.000000e+00
##
## NOTE: The total number of operators greater than population size
## NOTE: I'm increasing the population size to 10 (operators+1).
##
## Data Type: Floating Point
## Operators (code number, name, population)
##  (1) Cloning.........................  0
##  (2) Uniform Mutation................  1
##  (3) Boundary Mutation...............  1
##  (4) Non-Uniform Mutation............  1
##  (5) Polytope Crossover..............  1
##  (6) Simple Crossover................  2
##  (7) Whole Non-Uniform Mutation.....  1
##  (8) Heuristic Crossover.............  2
##  (9) Local-Minimum Crossover.........  0
##
## HARD Maximum Number of Generations: 12
## Maximum Nonchanging Generations: 2
## Population size       : 10
## Convergence Tolerance: 1.000000e-21
```

```
EI(xin, gp2)
```

```
## [1] 0.052808
```

```
xs <- sort(c(seq(0, 1, length = 100), xi))
gpp <- predict(gp2, newdata = xs, type = "SK")
plot(xs, gpp$mean, ylim = c(-2, 2), type = "l", col = "red", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
points(xi, y, pch = 4,lwd = 4, col = "blue")
lines(xs, gpp$upper95, lty = 2, lwd = 3)
lines(xs, gpp$lower95, lty = 2, lwd = 3)
abline(v = xin)
plot(xs, sapply(xs, EI, model = gp2), type = "l", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
```



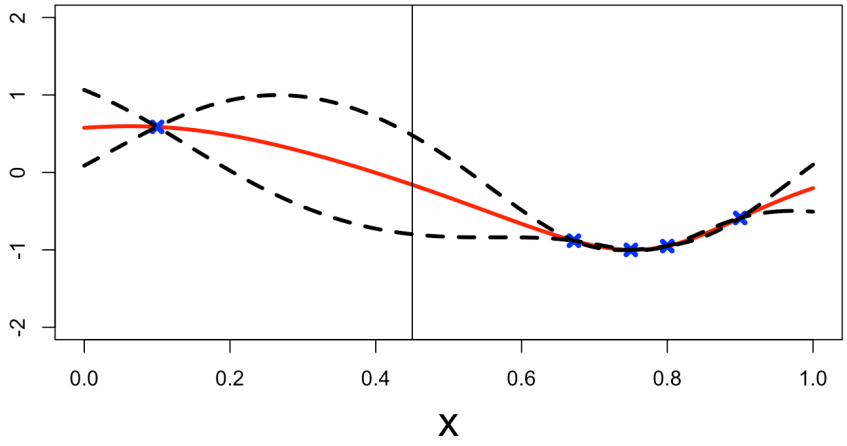


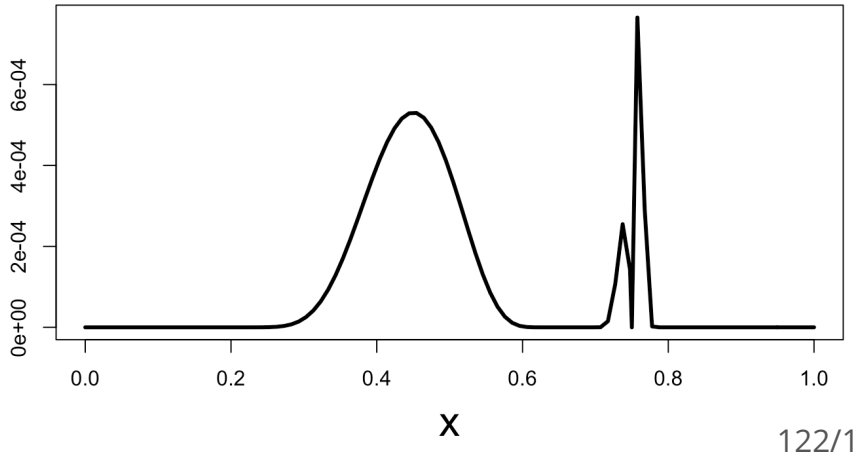
```
xi <- rbind(xi, xin)
y <- c(y, fn(xin))
gp3 <- km(design = xi, response = y, control = list(trace = F))
xin <- max_EI(model = gp3, lower = 0, upper = 1, control = list(trace = F))$par


##
##
## Mon Aug 26 15:55:26 2019
## Domains:
##  0.000000e+00   <=  X1   <=    1.000000e+00
##
## NOTE: The total number of operators greater than population size
## NOTE: I'm increasing the population size to 10 (operators+1).
##
## Data Type: Floating Point
## Operators (code number, name, population)
##  (1) Cloning.........................  0
##  (2) Uniform Mutation................  1
##  (3) Boundary Mutation...............  1
##  (4) Non-Uniform Mutation............  1
##  (5) Polytope Crossover..............  1
##  (6) Simple Crossover................  2
##  (7) Whole Non-Uniform Mutation.....  1
##  (8) Heuristic Crossover............  2
##  (9) Local-Minimum Crossover........  0
##
## HARD Maximum Number of Generations: 12
## Maximum Nonchanging Generations: 2
## Population size       : 10
## Convergence Tolerance: 1.000000e-21
```

```
EI(xin, gp3)
```

```
## [1] 0.0005309279
```

```
xs <- sort(c(seq(0, 1, length = 100), xi))
gpp <- predict(gp3, newdata = xs, type = "SK")
plot(xs, gpp$mean, ylim = c(-2, 2), type = "l", col = "red", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
points(xi, y, pch = 4,lwd = 4, col = "blue")
lines(xs, gpp$upper95, lty = 2, lwd = 3)
lines(xs, gpp$lower95, lty = 2, lwd = 3)
abline(v = xin)
plot(xs, sapply(xs, EI, model = gp3), type = "l", lwd = 3, ylab = "", xlab = "x", cex.lab = 2)
```





Sensitivity analysis

To analyse the relative importance of different model inputs, functional analysis of variance (FANOVA) can be used to decompose $g(\boldsymbol{x})$ into an additive form (see Saltelli et al. 2008)

Assume $x_i \sim w_i(x)$, eg the input variables may be normally distributed or uniformly distributed. Then

$$g(\boldsymbol{x}) = \mu_0 + \sum_{i=1}^{k} \mu_i(\boldsymbol{x}) + \sum_{i>j} \sum \mu_{ij}(\boldsymbol{x}) + \ldots + \mu_{12\cdots k}(\boldsymbol{x})$$

where

$$\mu_0 = \int g(\boldsymbol{x}) \prod_i w(x_i)\, \mathrm{d}(\boldsymbol{x})\,, \qquad \mu_i(\boldsymbol{x}) = \int g(\boldsymbol{x}) \prod_{j\neq i} w_j(x_j)\, \mathrm{d}x_j - \mu_0\,,$$

and

$$\mu_{ij}(\boldsymbol{x}) = \int g(\boldsymbol{x}) \prod_{l\neq i,j} w_l(x_l)\, \mathrm{d}x_l - \mu_i(\boldsymbol{x}) - \mu_j(\boldsymbol{x}) + \mu_0$$

with higher-order terms defined similarly

A (normalised measure) of the impact of each variable can be obtained by assessing the variance of each μ_i term via a **Sobol' index**:

$$S_i = \frac{\mathrm{Var}_{x_i}\left\{\mu_i(\boldsymbol{x})\right\}}{\mathrm{Var}_{\boldsymbol{x}}\left\{g(\boldsymbol{x})\right\}} \in [0,1]$$

The impact of higher-order terms can be assessed via the **total variance index**:

$$T_i = 1 - \frac{\mathrm{Var}_{\boldsymbol{x}_{(i)}}\left\{\mu_{-i}(\boldsymbol{x})\right\}}{\mathrm{Var}_{\boldsymbol{x}}\left\{g(\boldsymbol{x})\right\}} \in [0,1]$$

where $\boldsymbol{x}_{(i)} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)^{\mathrm{T}}$ and

$$\mu_{-i}(\boldsymbol{x}) = \int g(\boldsymbol{x}) w_i(x_i)\, \mathrm{d}x_i - \mu_0\,,$$

S_i measures the "main effect" of the ith variable, and T_i measures it's total effect including through "interactions" with other variables

- big differences between S_i and T_i suggest the ith variable interacts with other variables

The sensitivity indices can be estimated using Monte Carlo methods (essentially nested MC to estimate the expectations and variances)

- for computationally expensive models, sample from a GP emulator

Various methods are implemented in the sensitivity package

EBM example:

```
library(sensitivity)
d <- 2; n <- 1000
X1 <- data.frame(matrix(runif(d * n), nrow = n))
X2 <- data.frame(matrix(runif(d * n), nrow = n))
colnames(X1) <- colnames(ebm)[2:3]; colnames(X2) <- colnames(ebm)[2:3]
res <- sobolGP(model = gpebm, type = "UK", MCmethod = "sobol2002", X1, X2)
res
```

```
##
## Method: sobol2002
##
## Model runs: 20
##
## Number of GP realizations: 100
##
## Kriging type: UK
##
##      estimate   std. error  min. c.i.  max. c.i.
## S1 0.09603572 0.0006806932 0.09478208 0.09728398
## S2 0.98825845 0.0007460611 0.98702092 0.98961288
##
##      estimate   std. error  min. c.i.  max. c.i.
## T1 0.01232153 0.0005711950 0.01126854 0.01360374
## T2 0.93075208 0.0005361857 0.92958659 0.93178356
```

Uncertainty quantification

Computer experiments are an important statistical contribution to the field of **uncertainty quantification (UQ)**

- interdisciplinary topic on the interface of Statistics and Applied Maths
- methodologies for taking account of uncertainties when mathematical and computer models are used to describe real-world phenomena

Space-filling designs and (GP) emulators are very general, and can be applied to a variety of **black box** computer models

- typically require a lot less knowledge about the model than alternative methods from numerical analysis (although at some loss of efficiency)

GP emulators can be used as priors for Bayesian calibration of computer models (Kennedy and O'Hagan 2001)

- eg learning tuning parameters (cf parameter estimation, albeit with various important nuances around interpretation and physical understanding)
- data fusion: combining computer model runs and data from real experiments

References

Atkinson, A. C., A. N. Donev, and R. D. Tobias. 2007. *Optimum Experimental Design, with Sas*. 2nd ed. Oxford: Oxford University Press.

Basu, D. 1980. “Randomization Analysis of Experimental Data: The Fisher Randomization Test.” *Journal of the American Statistical Association* 75: 575–82.

Beck, L., B. Mansour Dia, L. F. R. Espath, Q. Long, and R. Tempone. 2018. “Fast Bayesian Experimental Design: Laplace-Based Importance Sampling for the Expected Information Gain.” *Computer Methods in Applied Mechanics and Engineering* 334: 523–53.

Box, G. E. P., and R. D. Meyer. 1986. “An Analysis of Unreplicated Fractional Factorials.” *Technometrics* 28: 11–18.

Chaloner, K., and I. Verdinelli. 1995. “Bayesian Experimental Design: A Review.” *Statistical Science* 10: 273–304.

Cox, D. R., and N. Reid. 2000. *The Theory of the Design of Experiments*. Boca Raton: Chapman; Hall/CRC Press.

Cuthbert, D. 1959. “Use of Half-Normal Plots in Interpreting Factorial Two-Level Experiments.” *Technometrics* 1: 311–41.

Dasgupta, T., N. S. Pillai, and D. B. Rubin. 2015. “Causal Inference from 2^k Factorial Designs by Using Potential Outcomes.” *Journal of the Royal Statistical Society B* 77: 727–53.

Fang, K.-T., R. Li, and A. Sudjianto. 2006. *Design and Modelling for Computer Experiments*. Boca Raton: Chapman; Hall/CRC Press.