Design of Experiments

Arnaud Legrand and Jean-Marc Vincent

Scientific Methodology and Performance Evaluation

Key concepts

There are two key concepts:

replication and randomization

You replicate to increase reliability. You randomize to reduce bias.

If you replicate thoroughly and randomize properly,
you will not go far wrong.

Key concepts

There are two key concepts:

replication and randomization

You replicate to increase reliability. You randomize to reduce bias.

If you replicate thoroughly and randomize properly, you will not go far wrong.

It doesn't matter if you cannot do your own advanced statistical analysis. If you designed your experiments properly, you may be able to find somebody to help you with the statistics.

If your experiments are not properly designed, then no matter how good you are at statistics, your experimental efforts will have been in vain.

Not even the most sophisticated statistical analysis can turn a flawed experiment into a good one.

Other important concepts:

- Pseudo-replication
- Experimental vs. observational data

Select the problem to study

Clearly define the kind of system to study, the kind of phenomenon to observe (state, evolution of state through time), the kind of study to conduct (descriptive, exploratory, prediction, hypothesis testing, . . .)

This is quite important as the set of experiments to perform will be completely different when you are:

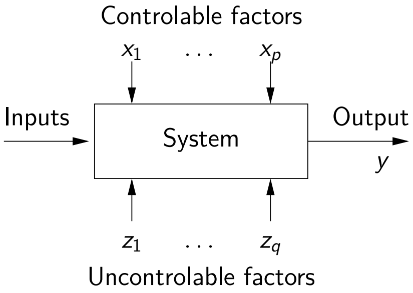
- studying the stabilization of a peer-to-peer algorithm under a high churn
- trying to compare various scheduling algorithms or code versions
- modeling the response time of a server under a workload close to the server saturation
- . . .

This step will help you to determine which kind of experiment design you should use.

Define the set of relevant *response*

The system under study is generally modeled through a black-box model:

- some output variable/response(y)
- some inputs are fully unknown
- some input variables ($x_1, \ldots, x_p$) are controllable
- whereas some others ($z_1, \ldots, z_q$) are uncontrollable



In our case, the response could be:

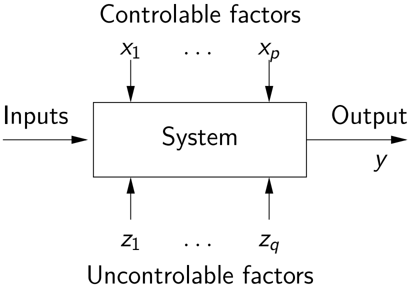
- the makespan of a scheduling algorithm
- the amount of messages exchanged in a peer-to-peer system
- the convergence time of distributed algorithm
- the average length of a random walk
- the amount of energy or of memory used

Some of these metrics are the result of complex aggregation of measurements so they should be carefully recorded to check their correctness

Determine the set of relevant *factors* or *variables*

The system under study is generally modeled though a black-box model:

- some output variable/response(y)
- some inputs are fully unknown
- some input variables ($x_1,\ldots,x_p$) are controllable
- whereas some others ($z_1, \ldots, z_q$) are uncontrollable



Typical controllable variables could be:

- the heuristic used (e.g., FIFO, HEFT, ...)
- one of their parameters (e.g., replication factor, a threshold, ...)
- the size of the platform
- the degree of heterogeneity
- the version of the compiler

Uncontrollable variables could be:

- temperature, humidity, moon phase, road surface conditions
- someone using the machine and interfering with the experiment

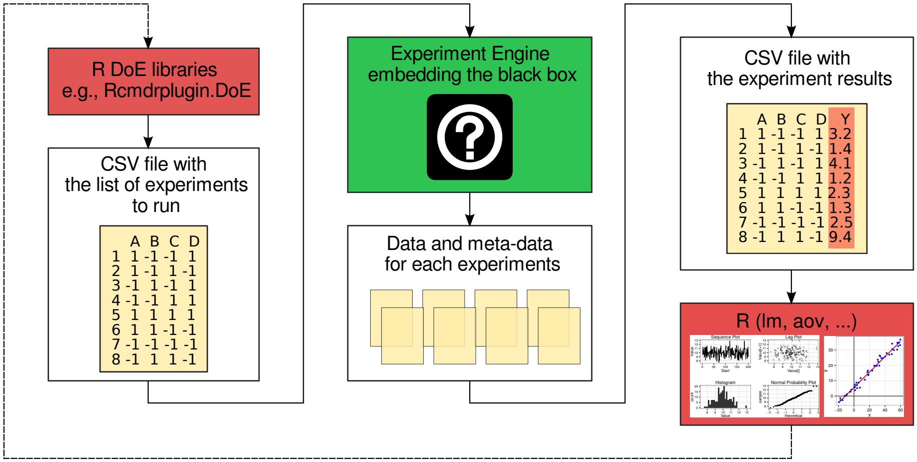
You should carefully record all the factors you can think of

Typical case studies

The typical case studies defined in the first step could include:

- Determining which variables are most influential on the response y (factorial designs, screening designs, analysis of variance)
  - Allows to distinguish between primary factors whose influence on the response should be modeled and secondary factors whose impact should be averaged
  - Allows to determine whether some factors interact in the response
- Devise an analytical model of the response y as a function of the primary factors x (regression, lhs designs)
- Fit a an analytical model (regression, response surface methodology, optimal designs)
  - Can then be used to determine where to set the primary factors x so that response y is always close to a desired value or is minimized/maximized
- Determining where to set the primary factors x so that variability in response y is small i.e., so that the effect of uncontrollable variables $z_1, \ldots, z_q$ is minimized (robust designs, Taguchi designs)

General Workflow



	A	B	C	D
1	1	-1	-1	1
2	1	-1	1	-1
3	-1	1	-1	1
4	-1	-1	1	1
5	1	1	1	1
6	1	1	-1	-1
7	-1	-1	-1	-1
8	-1	1	1	-1

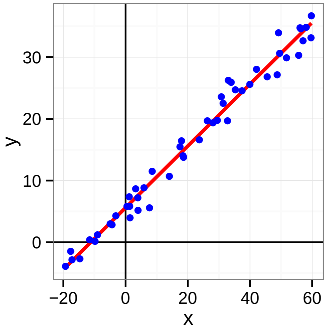
	A	B	C	D	Y
1	1	-1	-1	1	3.2
2	1	-1	1	-1	1.4
3	-1	1	-1	1	4.1
4	-1	-1	1	1	1.2
5	1	1	1	1	2.3
6	1	1	-1	-1	1.3
7	-1	-1	-1	-1	2.5
8	-1	1	1	-1	9.4

Outline

Linear Regression

$$Y = a + bX + \varepsilon$$

- Y is the response variable
- X is a continuous explanatory variable
- a is the intercept
- b is the slope
- ε is some noise



When there are 2 explanatory variables:

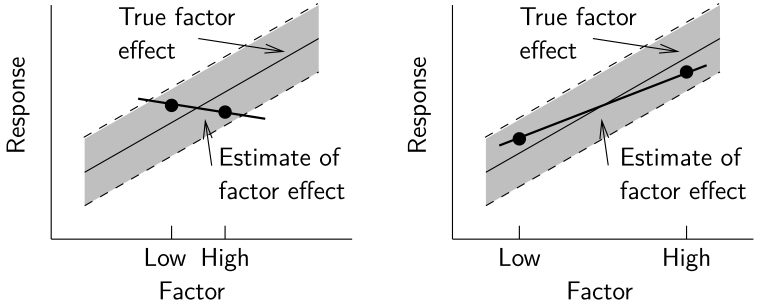
$$Y = a + b^{(1)}X^{(1)} + b^{(2)}X^{(2)} + b^{(1,2)}X^{(1)}X^{(2)} + \varepsilon$$

ε is generally assumed to be independent of $X^{(k)}$, hence it needs to be checked once the regression is done

- Although your phenomenon is not linear, the linear model helps for initial investigations (as a first crude approximation)
- You should always wonder whether there is a way of looking at your problem where it is linear

2-level factorial designs

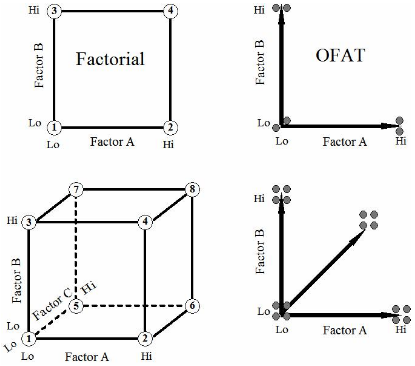
1. Decide a low and a high value for



   The different values are by convention encoded with -1 and 1 but these are not *real* numbers

2. Test every (2^p) combination of high and low values, possibly replicating for each combination.
   By varying everything, we can detect interactions right away

The downsides of the *One Factor At a Time* approach



- Only a very small fraction of the space is covered (bias) ☹
- Interaction between factors cannot be estimated ☹
- Each replication allows to improve the estimation quality of only one factor, hence it requires more runs to have good estimates of all factors ☹

Unless dealing with a very simple problem, it is always better to change parameters all together than change parameters One Factor at a Time

Generating a 2^p Design

```r
library(FrF2)
d1 = FrF2(nruns=8 ,nfactors=3 , blocks=1 , replications = 2,
         randomize= TRUE, seed= 26052 ,
         factor.names=list(A=c(-1,1), B=c(-1,1), C=c(-1,1))); d1 ;
```

```
 creating full factorial with 8 runs ...

   run.no run.no.std.rp  A  B  C
1       1           2.1  1 -1 -1
2       2           6.1  1 -1  1
3       3           3.1 -1  1 -1
4       4           5.1 -1 -1  1
...
15     15           1.2 -1 -1 -1
16     16           4.2  1  1 -1
class=design, type= full factorial
NOTE: columns run.no and run.no.std.rp are annotation, not part of t
```

How can we analyze something like this?

Outline

What if my number of factors is large ?

If $p = 8$, and the global variability is large, we may have to do $r = 5$ replications, hence $2^p.r = 256 \times 5 = 1280$ experiments!!!

- Then, you need something intermediate between OFAT and a full factorial 2^p design.
- It probably does not really make sense to study the joint effect of changing A, B, C, D, E, F, G, and H at the same time...

You should then go for a fractional 2^{p-k} design that will still make sure the combinations are well spread and the design is well balanced

Fractional designs

```
d2 = FrF2(nruns=8 ,nfactors=4 , blocks=1 , replications = 2,
         randomize= TRUE, seed= 26052 ,
         factor.names=list(A=c(-1,1), B=c(-1,1), C=c(-1,1), D=c(-1,1
d2 ;
```

```
   run.no run.no.std.rp  A  B  C  D
1       1           2.1  1 -1 -1  1
2       2           6.1  1 -1  1 -1
3       3           3.1 -1  1 -1  1
4       4           5.1 -1 -1  1  1
5       5           8.1  1  1  1  1
...
13     13           2.2  1 -1 -1  1
14     14           5.2 -1 -1  1  1
15     15           1.2 -1 -1 -1 -1
16     16           4.2  1  1 -1 -1
class=design, type= FrF2
NOTE: columns run.no and run.no.std.rp are annotation,
      not part of the data frame
```

Not much gain here... Fractional designs have constraints but allow you to control how much you loose

Saving a lot of time/money: Plackett-Burman screening designs

```r
d3 <-  pb(nruns= 20 ,n12.taguchi= FALSE ,nfactors= 20 -1, ncenter= 0 ,
    replications= 1 ,repeat.only= FALSE ,randomize= TRUE ,seed= 26654 ,
    factor.names=list( A=c(-1,1),B=c(-1,1),C=c(-1,1),D=c(-1,1),
     E=c(-1,1),F=c(-1,1),G=c(-1, 1),H=c(-1,1),J=c(-1,1),K=c(-1,1),
     L=c(-1,1),M=c(-1,1),N=c(-1,1),O=c(-1,1),P=c(-1,1) ) ) ; d3
```

```
    A  B  C  D  E  F  G  H  J  K  L  M  N  O  P e1 e2 e3 e4
1   1  1  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  1 -1  1
3  -1 -1 -1 -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  1 -1  1
5  -1 -1  1  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 -1  1  1
7   1  1 -1  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 -1 -1 -1
9   1 -1  1 -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  1  1 -1
11 -1  1  1  1  1 -1  1 -1  1 -1 -1 -1 -1  1  1 -1  1  1 -1
12  1  1  1  1 -1  1 -1  1 -1 -1 -1 -1  1  1 -1  1  1 -1 -1
13 -1 -1 -1  1  1 -1  1  1 -1 -1  1  1  1  1 -1  1 -1  1 -1
14  1 -1 -1 -1 -1  1  1 -1  1  1 -1 -1  1  1  1  1 -1  1 -1
15  1 -1  1  1 -1 -1  1  1  1  1 -1  1 -1  1 -1 -1 -1 -1  1
16  1  1 -1  1 -1  1 -1 -1 -1 -1  1  1 -1  1  1 -1 -1  1  1
17 -1  1 -1  1 -1 -1 -1 -1  1  1 -1  1  1 -1 -1  1  1  1  1
```

Only allows to estimate primary factors, not interations

Preliminary step for further investigation

Outline

What about having more than two levels?

Before even considering the generation, how would this analyzed?

- ANOVA still works and interpretation is OK when there are one (1-way ANOVA) or two (2-way ANOVA) factors (with several levels)
- Otherwise, it is a nightmare to analyze and you should decrease either the number of factors or the number of levels

In term of design, you can still go for all combinations

General Full Factorial Experiments

```r
d4 <- fac.design(nfactors= 2 ,replications= 3 ,repeat.only= FALSE ,
                 blocks= 1 , randomize= TRUE ,seed= 17366 ,
                 nlevels=c( 3,5 ), factor.names=list(
                 Size=c("S","M","L"),Color=c("R","G","B","M","Y"))) ; d4
```

```
 creating full factorial with 15 runs ...

   run.no run.no.std.rp Size Color
1       1           6.1    L     G
2       2          10.1    S     M
3       3          12.1    L     M
4       4           2.1    M     R
5       5          11.1    M     M
6       6          14.1    M     Y
7       7           4.1    S     G
8       8           3.1    L     R
9       9          13.1    S     Y
10     10          15.1    L     Y
11     11           5.1    M     G
12     12           1.1    S     R
13     13           8.1    M     B
14     14           7.1    S     B
15     15           9.1    L     B
16     16           1.2    S     R
17     17           8.2    M     B
```

Reducing the size of such designs

You can still sample from it but the outcome is likely to be not well balanced

- $\rightsquigarrow$ the estimation may not be that good and probably quite biased because of this ☹

```
d4[sample(size=5,replace=FALSE,1:nrow(d4)),]
```

```
   Size Color
29    L     G
30    L     M
41    L     G
3     L     M
25    S     B
```

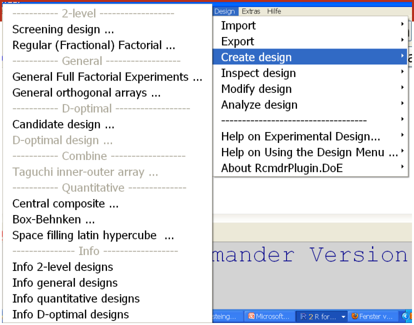
That's why you should try to reduce as much as possible the number of factors and of levels if you can

How do you expect me to ever remember all these commands ?

For the R commands, there is a trick: 😉

Use Rcmdr and RcmdrPlugin.DoE
(by Ulrike Grömping)

Simply `library(RcmdrPlugin.DoE)...`



----------- 2-level -----------------
Screening design ...
Regular (Fractional) Factorial ...
----------- General -----------------
General Full Factorial Experiments ...
General orthogonal arrays ...
----------- D-optimal -----------------
Candidate design ...
D-optimal design ...
----------- Combine -----------------
Taguchi inner-outer array ...
----------- Quantitative -------------
Central composite ...
Box-Behnken ...
Space filling latin hypercube ...
-------------- Info -----------------
Info 2-level designs
Info general designs
Info quantitative designs
Info D-optimal designs
Design
Extras
Hilfe
Import
Export
Create design
Inspect design
Modify design
Analyze design
Help on Experimental Design...
Help on Using the Design Menu ...
About RcmdrPlugin.DoE
mander Version

You should only remember the principles and try to understand the underlying hypothesis

- ANOVA enables to discriminate real effects from noise in factorial experiments. *It relies on homoscedasticity and normality (or requires large number of samples)*
- 2-level factorial designs are a simple way to go and are more efficient than OFAT experiments
- Replicate thoroughly and randomize properly: you will not go far wrong

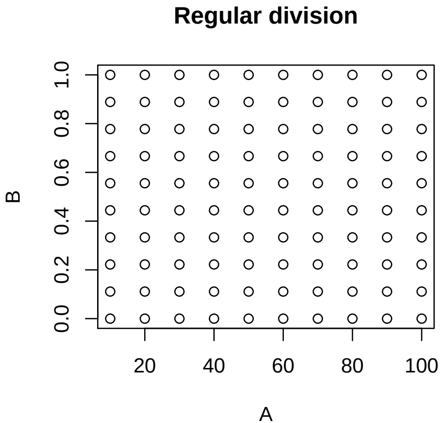
Outline

Without any information about the response

Then we should not favor a region over an other

- What about all combinations of a regular division?

```r
x <- seq(10, 100, length.out = 10)
y <- seq(0, 1, length.out = 10)
d5_regular <- expand.grid(A = x, B = y)
plot(d5_regular, main="Regular division")
```

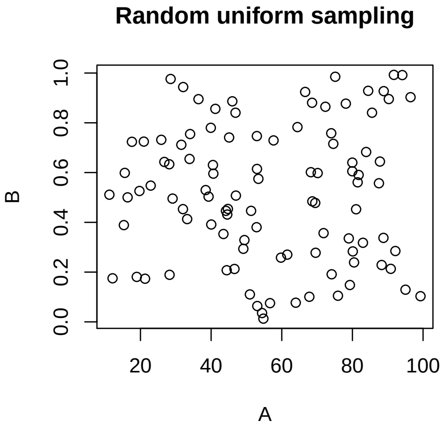


Can we have a less biased design?

We should not favor any particular value

- What about a uniform sampling then?

```r
set.seed(1);
x <- runif(100,min=10,max=100); y <- runif(100, min=0,max=1)
d5_unif <- data.frame(A = x, B = y)
plot(d5_unif, main="Random uniform sampling")
```

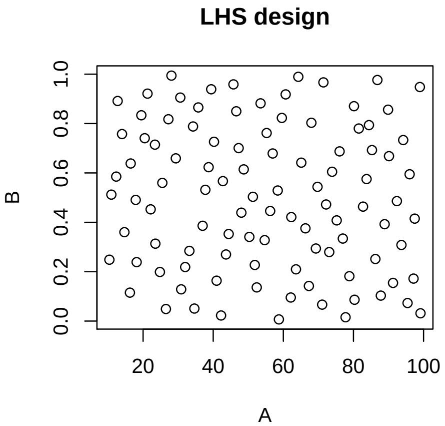


Can we have a design covering better the whole space?

We do not want to miss any region

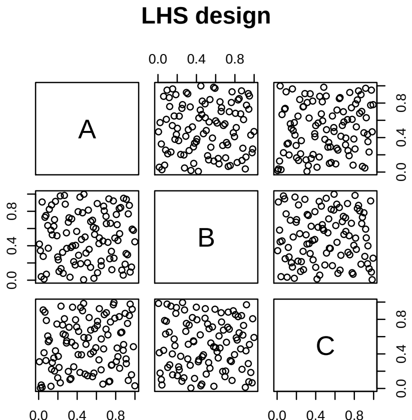
- Space filling designs: Latin Hyper Square designs and the maximin criteria

```r
library(DoE.wrapper)
d5_maximin <- lhs.design( type= "maximin" , nruns= 100 ,nfactors= 2 ,
  digits= NULL , seed= 27041 , factor.names=list( A=c(10,100),B=c(0,1) ) )
plot(d5_maximin , select = c( "A","B" ), main="LHS design")
```



This still reasonably works in higher dimensions

```r
library(DoE.wrapper);   set.seed(42);
d5_HD = lhs.design( type= "maximin" , nruns= 100 ,nfactors= 3 ,
    seed= 42 , factor.names=list( A=c(0,1),B=c(0,1),C=c(0,1) ) )
Response5 = 10 + 2*as.numeric(d5_HD$A) + 3*as.numeric(d5_HD$B)*as.numeric(d5_HD$C) +
    rnorm(nrow(d5_HD),sd=1)
d5_HD <- add.response(d5_HD, Response5, replace=TRUE)
plot(d5_HD , select = c( "A","B","C" ), main="LHS design")
```



What about the analysis?

```r
summary(lm(Response5 ~ (A + B + C)^2, data = d5_HD))
```

```
Call:
lm.default(formula = Response5 ~ (A + B + C)^2, data = d5_HD)

Residuals:
     Min       1Q   Median       3Q      Max
-2.90043 -0.64768  0.00095  0.75471  2.61620

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.0932     0.5920  17.049   <2e-16 ***
A             1.5542     0.9686   1.605   0.1120
B             1.1188     0.8904   1.257   0.2121
C            -1.4085     0.9283  -1.517   0.1326
A:B          -2.3379     1.3228  -1.767   0.0804 .
A:C           3.0344     1.2428   2.442   0.0165 *
B:C           2.9668     1.2910   2.298   0.0238 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1

Residual standard error: 1.087 on 93 degrees of freedom
Multiple R-squared:  0.451,	Adjusted R-squared:  0.4156
F-statistic: 12.74 on 6 and 93 DF,  p-value: 1.909e-10
```

There is actually too much variability to conclude anything here (look at the R^2)

We know from the anova that B:C is significant but its Std. Error is still 1.29

We should add another round of 3 times more experiments to halve it

What happens if we fit a simpler model ?

```r
summary(lm(Response5 ~ A + B:C, data = d5_HD))
```

```
Call:
lm.default(formula = Response5 ~ A + B:C, data = d5_HD)

Residuals:
     Min       1Q   Median       3Q      Max
-3.00860 -0.71419 -0.00565  0.74843  2.98579

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.0054     0.2471  40.489  < 2e-16 ***
A             1.8262     0.3920   4.659 1.01e-05 ***
B:C           3.0066     0.5247   5.730 1.13e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.119 on 97 degrees of freedom
Multiple R-squared:  0.3938,	Adjusted R-squared:  0.3814
F-statistic: 31.51 on 2 and 97 DF,  p-value: 2.852e-11
```

The Std. Errors decreased but remain quite high

As one could expect, the R^2 has decreased... ☹

Let's cheat. . . 😉

```r
Response5 = 10 + 2*as.numeric(d5_HD$A) +  3*as.numeric(d5_HD$B)*as.numeric(d5_HD$C) +
    rnorm(nrow(d5_HD),sd=.2) # Decreasing variability
d5_HD <- add.response(d5_HD, Response5, replace=TRUE)
summary(lm(Response5 ~ (A + B + C)^2, data = d5_HD))
```

```
Call:
lm.default(formula = Response5 ~ (A + B + C)^2, data = d5_HD)
Residuals:
     Min       1Q   Median       3Q      Max
-0.50030 -0.10491 -0.00945  0.13446  0.47068
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.06454    0.10992  91.558  < 2e-16 ***
A            1.58630    0.17986   8.820 6.41e-14 ***
B            0.13805    0.16533   0.835   0.4059
C            0.09524    0.17236   0.553   0.5819
A:B          0.46421    0.24562   1.890   0.0619 .
A:C          0.30745    0.23078   1.332   0.1860
B:C          2.33722    0.23972   9.750 6.92e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2019 on 93 degrees of freedom
Multiple R-squared:  0.9551,        Adjusted R-squared:  0.9522
F-statistic: 329.9 on 6 and 93 DF,  p-value: < 2.2e-16
```

One should actually instead fit the simple model suggested by the previous analysis:

`y~A+B:C`

You should use parsimony both in experiment design and modeling

Parsimony (1/2)

```r
summary(lm(Response5 ~ A + B:C, data = d5_HD))
```

```
Call:
lm.default(formula = Response5 ~ A + B:C, data = d5_HD)

Residuals:
     Min       1Q   Median       3Q      Max
-0.56483 -0.11393  0.00626  0.12994  0.46614

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.05536    0.04609  218.18   <2e-16 ***
A            1.94985    0.07311   26.67   <2e-16 ***
B:C          2.90476    0.09786   29.68   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2087 on 97 degrees of freedom
Multiple R-squared:   0.95,       Adjusted R-squared:  0.949
F-statistic: 921.8 on 2 and 97 DF,  p-value: < 2.2e-16
```

Parsimony (2/2)

The principle of parsimony is attributed to the 14th century English philosopher William of Occam:

> *"Given a set of equally good explanations for a given phenomenon, the correct explanation is the simplest explanation"*

Parsimony (2/2)

The principle of parsimony is attributed to the 14th century English philosopher William of Occam:

> *"Given a set of equally good explanations for a given phenomenon, the correct explanation is the simplest explanation"*

- Models should have as few parameters as possible
- Linear models should be preferred to non-linear models
- Models should be pared down until they are *minimal adequate*

Parsimony (2/2)

The principle of parsimony is attributed to the 14th century English philosopher William of Occam:

> *"Given a set of equally good explanations for a given phenomenon, the correct explanation is the simplest explanation"*

- Models should have as few parameters as possible
- Linear models should be preferred to non-linear models
- Models should be pared down until they are *minimal adequate*

This means, a variable should be retained in the model only if it causes a significant increase in deviance when removed from the current model

> *A model should be as simple as possible. But no simpler.*
>
> *– A. Einstein*

Outline

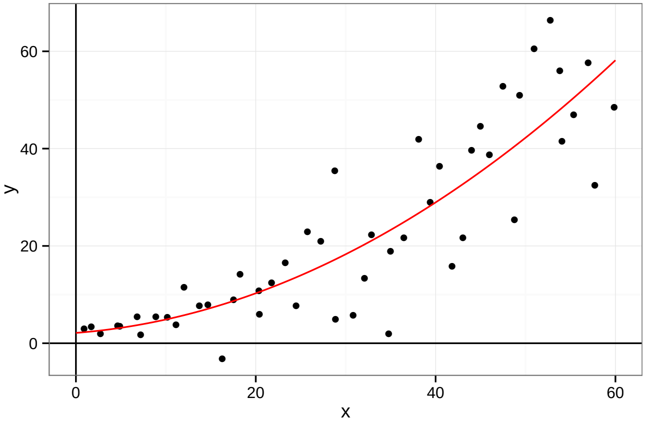
Working out a toy example

```r
x=lhs.design(type= "maximin", nruns=50, nfactors=1, seed=77,
             factor.names=list(x=c(0,60)))$x
y=3+x^2/60 + x*rnorm(length(x),sd=.3)
df = data.frame(x=x,y=y)
reg_quad <- lm(data=df,y~x+I(x^2))
summary(reg_quad)
```

```
Call:
lm.default(formula = y ~ x + I(x^2), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-21.7802  -4.5247   0.7544   5.1195  20.0284

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.124017   4.007473   0.530   0.5986
x           0.143694   0.310362   0.463   0.6455
I(x^2)      0.013169   0.005021   2.623   0.0117 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.483 on 47 degrees of freedom
Multiple R-squared:  0.7647,        Adjusted R-squared:  0.7547
F-statistic: 76.36 on 2 and 47 DF,  p-value: 1.715e-15
```

We can clearly see where the heteroscedasticity comes from

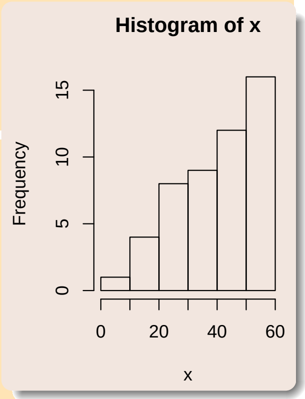
```r
xv <- seq(0,60,.5)
yv <- predict(reg_quad,list(x=xv,x2=xv^2))
ggplot(data=df, aes(x=x,y=y)) + theme_bw() +
    geom_hline(yintercept=0) + geom_vline(xintercept=0) +
    geom_point(aes(x=x,y=y)) +
    geom_line(data=data.frame(x=xv,y=yv),aes(x=x,y=y),color="red")
```



Adding more points where there is more variability

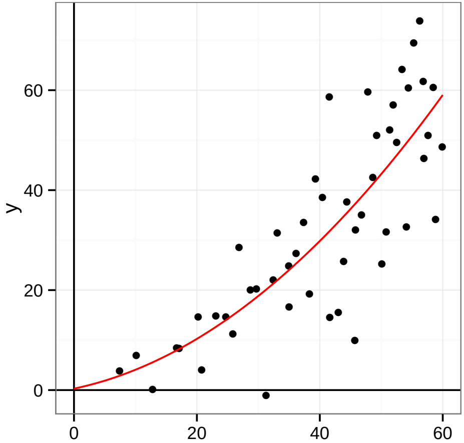
```r
x=sqrt(lhs.design( type= "maximin" , nruns= 50 ,nfactors= 1 ,
       seed= 77 , factor.names=list( x=c(0,60^2) ) )$x)
y=3+x^2/60 + x*rnorm(length(x),sd=.3)

df = data.frame(x=x,y=y)
reg_quad <- lm(data=df,y~x+I(x^2))
summary(reg_quad)
```

```
Call:
lm.default(formula = y ~ x + I(x^2), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-27.256  -7.269   1.143   7.702  26.904

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.292996  10.607678   0.028    0.978
x            0.257212   0.626398   0.411    0.683
I(x^2)       0.012031   0.008495   1.416    0.163

Residual standard error: 12.02 on 47 degrees of freedom
Multiple R-squared:  0.6569,      Adjusted R-squared:  0.6423
F-statistic: 44.99 on 2 and 47 DF,  p-value: 1.209e-11
```



Unfortunately, this does not really help

```r
xv <- seq(0,60,.5)
yv <- predict(reg_quad,list(x=xv,x2=xv^2))
ggplot(data=df, aes(x=x,y=y)) + theme_bw() +
    geom_hline(yintercept=0) + geom_vline(xintercept=0) +
    geom_point(aes(x=x,y=y)) +
    geom_line(data=data.frame(x=xv,y=yv),aes(x=x,y=y),color="red")
```



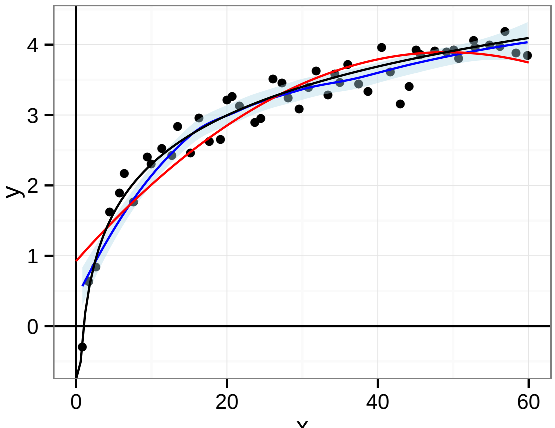
The R^2 will never exceed 0.66 because our model fails fully explaining variance

- We should thus rather replicate for large values of x and average the results
- The expected value will be the same but the variance will be reduced

Outline

What if even polynomial models seem inadequate?

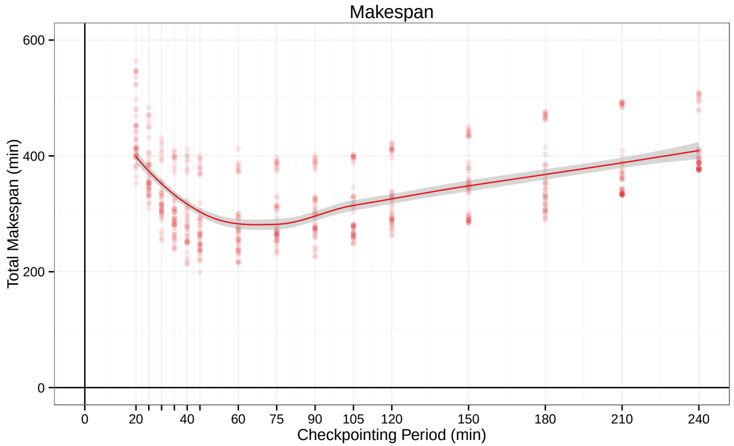
```
ggplot(data=df, aes(x=x,y=y)) + theme_bw() +
    geom_hline(yintercept=0) + geom_vline(xintercept=0) +
    geom_point(aes(x=x,y=y)) +
    stat_smooth(method="loess",color="blue",fill="lightblue") +
    geom_line(data=data.frame(x=xv,y=yv),aes(x=x,y=y),color="red") +
    stat_function(fun=log) # the true function
```



LOcal RegrESSion: builds on linear regression to locally fit a line or a polynom

This is a very biased *estimator* so use with care

Discuss about the shape



- "*the checkpointing period should be 68 minutes*": non-sense, uninteresting ☹
- "*optimality region is flat and one should rather overestimate the checkpointing period*" ☺

Outline

Remember what linear regression is about

$$Y = X.b + \varepsilon \text{ with } \varepsilon \sim \mathcal{N}_n(0_n, \sigma^2 I_n)$$

The least squares regression is the Best Linear Unbiased Estimate ($\mathrm{E}[\hat{\beta}] = b$ and $\mathrm{Var}(\hat{\beta})$ is minimal.

$$\hat{\beta} = (X^T.X)^{-1}.X^T.Y$$

- X is the model matrix
- $(X^T.X)$ is the covariance matrix

The variance of $\hat{\beta}$ (the uncertainty) is directly linked to $(X^T.X)$ whose eigenvalues should thus be as large as possible.

D optimality

When estimating model coefficients, it is intuitively better not to spread inputs but rather to use extreme values

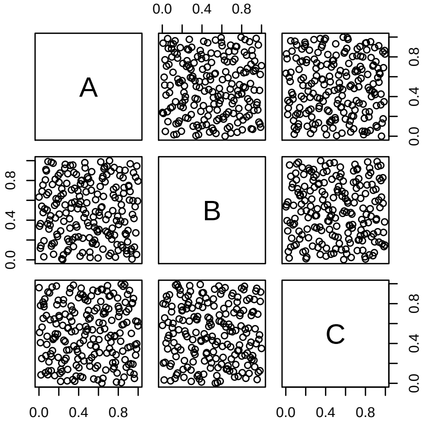
- Note: this approach assumes that the model is correct

This intuitive notion can be formalized for linear models (see Hoos):

- Minimize generalized variance of least squares estimates of model parameters (determinant of covariance matrix)
  $\rightsquigarrow$ D-optimal designs
- Minimize average variance (trace of covariance matrix)
  $\rightsquigarrow$ A-optimal designs
- Minimize average of predicted response over experimental region
  $\rightsquigarrow$ I-optimal designs

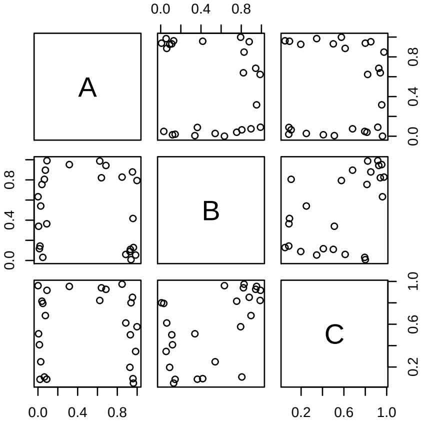
D-optimal Designs with R

```r
d7 <- lhs.design(type= "maximin", nruns= 200 , nfactors= 3,
                 digits=NULL, seed= 20521,
                 factor.names=list( A=c(0,1),B=c(0,1),C=c(0,1)))
d7.Dopt <- Dopt.design(25, data=d7, formula="~A +B:C", nRepeat= 5,
                       randomize= TRUE, seed=19583)
```



D-optimal Designs with R

```r
d7 <- lhs.design(type= "maximin", nruns= 200 , nfactors= 3,
                 digits=NULL, seed= 20521,
                 factor.names=list( A=c(0,1),B=c(0,1),C=c(0,1)))
d7.Dopt <- Dopt.design(25, data=d7, formula="~A +B:C", nRepeat= 5,
                       randomize= TRUE, seed=19583)
```



D-optimal designs

- Fractional designs and screening designs are D-optimal designs (with additional good properties) for 2-level factors and specific models
  - Full or almost full models (2^{n-k})
  - Models with no interaction at all (screening)
  - Algebraic methods exist for several very regular cases
- For arbitrary numbers of levels and specific models, it is possible to build optimal designs using ILP (exponential)
  - Each ILP variable correspond to a given configuration (exponential)
  - $\rightsquigarrow$ hardly usable unless the design space is really small
- Otherwise, use the Federov algorithm to select the "best" subset of k experiments among n "given" configurations
  - A greedy heuristic with restart
  - Quality can be difficult to evaluate for large design spaces

Conclusion

- Designing experiments can be fun! 😉
- Proceed carefully
  - The analysis is not simple but skilled statisticians can help you
  - The crucial part is actually the modeling, when you identify the factors, the response, and the kind of study
- This lecture only gives an overview but may already have changed your point of view on how to conduct experiments
- Remind the benefits of the sequential approach:
  - Parsimony
  - Use well-suited DoE and the corresponding analysis
  - Add measurements where there is variability

Recap on the lecture

1. Reproducibility is essential
   - literate programming with knitr or org-mode
   - laboratory notebook
2. Data manipulation and presentation
   - R, ggplot2, plyr, . . .
3. Introduction to probabilities and statistics
   - A probabilistic model allows you to assess the confidence of your claims
4. Linear regression
   - The linear model is quite general
   - This knowledge about the system allows you to improve estimates
5. Design of Experiments
   - Sequential approach
   - Designs/analysis suited to every study