Design of experiments

Gabriele Degola

January 2022

My page on the *shinyapp*: https://adaphetnodes.shinyapps.io/design_of_experiments/?user_d3318

```
library(FrF2)
library(DoE.wrapper)
library(tidyverse)
library(GGally)

seed <- 42
set.seed(seed)
```

Experimental design

As a first attempt, let's test a Plackett-Burman design on the 11 variables, that can take values between 0 and 1.100 runs are made.

```
pb <- pb(nruns=100, nfactors=11, n12.taguchi=FALSE, factor.names=list(x1=c(0,1),
↪  x2=c(0,1), x3=c(0,1), x4=c(0,1), x5=c(0,1), x6=c(0,1), x7=c(0,1), x8=c(0,1),
↪  x9=c(0,1), x10=c(0,1), x11=c(0,1)), seed=seed)
print(sum(is.na(pb)))
```

```
## [1] 0
```

```
write.table(x=pb, file="pb.csv", sep=",", row.names=FALSE, col.names=FALSE, quote=FALSE)
```

Let's analyse the results produced on the website.

```
pb_res <- read.csv(file="pb_res.csv") %>% select(-Date)
pb_res[pb_res[, "y"] < 0.1 & pb_res[, "y"] > -0.1, ]
```

```
##    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11           y
## 9   0  0  1  1  1  0  0  0  1   1   1 0.010758826
## 29  0  0  0  1  0  0  0  0  1   0   1 0.013491815
## 41  0  0  1  1  1  1  0  1  1   0   0 0.017516379
## 44  0  1  0  1  0  1  0  1  1   1   1 0.012031443
## 69  0  1  1  1  0  0  0  1  1   1   0 0.008057174
## 78  0  1  1  1  0  0  0  0  1   0   0 0.012095448
## 86  0  1  1  1  1  0  0  1  1   1   1 0.010653164
## 97  0  0  1  1  1  1  0  0  1   1   1 0.017507613
```

Some aspects of interest can be noticed when the response is 0 up to a noise. In particular, in the corresponding experiments the variables `x1`, `x4`, `x7` and `x9` always have the same values (0, 1, 0 and 1 respectively), while different values of the others do not considerably alter the results.

```
head(pb_res %>% arrange(desc(y)))
```

```
##    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11        y
## 1  0  1  1  1  1  0  0  1  0   0   1 2.019293
## 2  0  0  1  1  1  0  0  0  0   1   0 2.016155
## 3  0  1  0  1  0  1  0  0  0   1   0 2.014039
## 4  0  0  0  1  1  1  0  1  0   0   1 2.010210
## 5  0  0  0  1  1  1  0  0  0   1   1 2.008318
## 6  0  1  0  1  0  0  1  1  0   0   1 1.717260
```

```
head(pb_res %>% arrange(y))
```

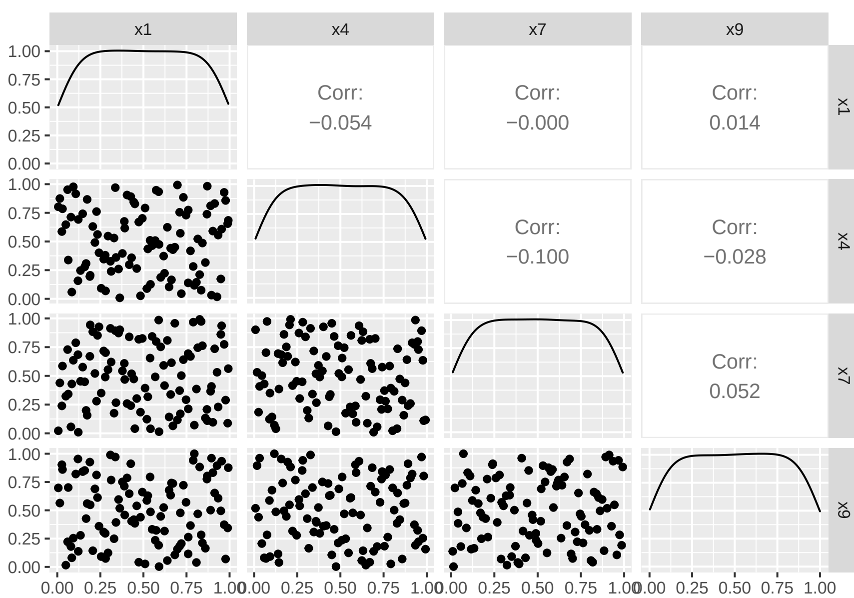
```
##    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11         y
## 1  1  0  1  0  0  1  0  1  1   1   1 -1.584767
## 2  1  0  1  0  1  0  1  1  1   0   1 -1.583495
## 3  1  1  1  0  0  0  1  0  1   1   0 -1.583088
## 4  1  0  1  0  1  0  0  1  1   1   1 -1.581281
## 5  1  0  1  0  1  1  0  0  1   1   0 -1.581214
## 6  1  1  1  0  0  0  0  1  1   0   0 -1.581111
```

It can be noticed that, apparently, highest results are obtained with `x1=0`, `x4=1` and `x9=0`. Trend is confirmed for the lowest results. `x7` appears to contribute less than the others, with no clear emerging pattern. However, we do not know anything about the underlying structure, so it would be wrong to assume linear contribution for the variables.

For next experiments, variables `x1`, `x4`, `x7` and `x9` are only taken into consideration. Let's try a *Latin Hyper Square* design on them.

```
lhs <- lhs.design(type="maximin", nruns=100, nfactors=11, seed=seed,
↪  factor.names=list(x1=c(0,1), x2=c(0,0), x3=c(0,0), x4=c(0,1), x5=c(0,0), x6=c(0,0),
↪  x7=c(0,1), x8=c(0,0), x9=c(0,1), x10=c(0,0), x11=c(0,0)))
lhs[is.na(lhs)] <- 0

ggpairs(lhs, columns=c("x1","x4","x7","x9"))
```




```
write.table(x=lhs, file="lhs.csv", sep=",", row.names=FALSE, col.names=FALSE,
↪  quote=FALSE)
```

Let's analyse the results produced on the website.

```
lhs_res <- read.csv(file="lhs_res.csv") %>% select(-Date)
head(lhs_res %>% arrange(desc(y)))
```

```
##          x1 x2 x3        x4 x5 x6        x7 x8         x9 x10 x11        y
## 1 0.6972074  0  0 0.9928581  0  0 0.1163498  0 0.15666176   0   0 3.287786
## 2 0.7099268  0  0 0.7549616  0  0 0.3718961  0 0.18287630   0   0 2.989595
## 3 0.7601092  0  0 0.7743564  0  0 0.2126829  0 0.26319609   0   0 2.882101
## 4 0.6386895  0  0 0.6238209  0  0 0.8083322  0 0.05627752   0   0 2.719800
## 5 0.6821239  0  0 0.4501121  0  0 0.9575589  0 0.10596519   0   0 2.694184
## 6 0.7594395  0  0 0.1389147  0  0 0.6936315  0 0.11476659   0   0 2.565348
```

```
head(lhs_res %>% arrange(y))
```

```
##   x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11         y
## 1  1  0  1  0  0  1  0  1  1   1   1 -1.584767
## 2  1  0  1  0  1  0  1  1  1   0   1 -1.583495
## 3  1  1  1  0  0  0  1  0  1   1   0 -1.583088
## 4  1  0  1  0  1  0  0  1  1   1   1 -1.581281
## 5  1  0  1  0  1  1  0  0  1   1   0 -1.581214
## 6  1  1  1  0  0  0  0  1  1   0   0 -1.581111
```

Regression models

With the results obtained so far, it is possible to fit regression model, in order to try to guess the underlying law. Let's start with a linear regression.

```
lin_reg1 <- lm(y ~ x1 + x4 + x7 + x9, data=lhs_res)
summary(lin_reg1)
```

```
##
## Call:
## lm.default(formula = y ~ x1 + x4 + x7 + x9, data = lhs_res)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7659 -0.4437 -0.1739  0.2021  1.6734
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.4200     0.1215  11.692  < 2e-16 ***
## x1           -0.2393     0.1117  -2.142   0.0334 *
## x4            0.7688     0.1118   6.878 8.04e-11 ***
## x7           -0.1629     0.1118  -1.457   0.1466
## x9           -2.0858     0.1117 -18.667  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6448 on 195 degrees of freedom
## Multiple R-squared:  0.6757, Adjusted R-squared:  0.6691
## F-statistic: 101.6 on 4 and 195 DF,  p-value: < 2.2e-16
```

According to the fitted model, variable x7 is not significant for predicting the response. Let's use two models in parallel, one including and one excluding it.

```r
lin_reg2 <- lm(y ~ x1 + x4 + x9, data=lhs_res)
summary(lin_reg2)
```

```
##
## Call:
## lm.default(formula = y ~ x1 + x4 + x9, data = lhs_res)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7525 -0.4017 -0.1999  0.2177  1.7052
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.3376     0.1078  12.411  < 2e-16 ***
## x1           -0.2392     0.1120  -2.136    0.034 *
## x4            0.7729     0.1121   6.897 7.15e-11 ***
## x9           -2.0879     0.1120 -18.634  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6467 on 196 degrees of freedom
## Multiple R-squared:  0.6722, Adjusted R-squared:  0.6672
## F-statistic:   134 on 3 and 196 DF,  p-value: < 2.2e-16
```

Now, let's generate a random example, feed it on the website and try to predict the response.

```r
test <- data.frame(matrix(0, ncol=11, nrow=2))
colnames(test) <- colnames(lhs)
test[1, c("x1","x4","x7","x9")] <- runif(4)
test[2, c("x1","x4","x7","x9")] <- runif(4)
test
```

```
##          x1 x2 x3        x4 x5 x6        x7 x8        x9 x10 x11
## 1 0.6500344  0  0 0.1901075  0  0 0.6931745  0 0.5437310   0   0
## 2 0.5776147  0  0 0.7346057  0  0 0.8512241  0 0.7153903   0   0
```

```r
write.table(test, file="", sep=",", row.names=FALSE, col.names=FALSE, quote=FALSE)
```

```
## 0.650034360820428,0,0,0.190107476897538,0,0,0.693174451123923,0,0.543731004698202,0,0
## 0.577614695765078,0,0,0.734605745412409,0,0,0.851224099285901,0,0.715390313183889,0,0
```

On the website, responses are ~ 1.49 and 1.14 respectively.

```r
predict.lm(lin_reg1, test)
```

```
##         1         2
## 0.1636095 0.2157516
```

```r
predict.lm(lin_reg2, test)
```

```
##         1         2
## 0.1937277 0.2734632
```

Mh, results are very poor.

Let's try to generalize fitting a quadratic regression model.

```r
quad_reg1 <- lm(y ~ poly(x1,2) + poly(x4,2) + poly(x7,2) + poly(x9,2), data=lhs_res)
summary(quad_reg1)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 2) + poly(x4, 2) + poly(x7,
##     2) + poly(x9, 2), data = lhs_res)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66131 -0.27925 -0.09252  0.22285  1.15776
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.560503   0.030234  18.539  < 2e-16 ***
## poly(x1, 2)1   -1.391413   0.428230  -3.249  0.00137 **
## poly(x1, 2)2   -7.017497   0.744359  -9.428  < 2e-16 ***
## poly(x4, 2)1    4.784700   0.429526  11.139  < 2e-16 ***
## poly(x4, 2)2   -0.162156   0.730885  -0.222  0.82466
## poly(x7, 2)1   -1.104446   0.428411  -2.578  0.01069 *
## poly(x7, 2)2    0.475995   0.701911   0.678  0.49850
## poly(x9, 2)1  -11.769355   0.428484 -27.467  < 2e-16 ***
## poly(x9, 2)2   -0.006144   0.764453  -0.008  0.99360
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4276 on 191 degrees of freedom
## Multiple R-squared:  0.8603, Adjusted R-squared:  0.8545
## F-statistic: 147.1 on 8 and 191 DF,  p-value: < 2.2e-16
```

```r
quad_reg2 <- lm(y ~ poly(x1,2) + poly(x4,2) + poly(x9,2), data=lhs_res)
summary(quad_reg2)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 2) + poly(x4, 2) + poly(x9,
##     2), data = lhs_res)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70323 -0.30324 -0.05422  0.18072  1.19130
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.56050    0.03063  18.299  < 2e-16 ***
## poly(x1, 2)1   -1.37674    0.43332  -3.177  0.00173 **
## poly(x1, 2)2   -6.80505    0.73321  -9.281  < 2e-16 ***
## poly(x4, 2)1    4.80550    0.43508  11.045  < 2e-16 ***
## poly(x4, 2)2   -0.03327    0.70927  -0.047  0.96264
## poly(x9, 2)1  -11.79233    0.43398 -27.172  < 2e-16 ***
## poly(x9, 2)2    0.06930    0.73819   0.094  0.92531
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.4332 on 193 degrees of freedom
## Multiple R-squared:  0.8551, Adjusted R-squared:  0.8506
## F-statistic: 189.9 on 6 and 193 DF,  p-value: < 2.2e-16
```

Results are interesting here! Apparently, x1^2 plays a significant role and, when it is considered, x7 is significant too. Resulting R^2 score is also higher than before.

```
quad_reg3 <- lm(y ~ poly(x1,2) + x4 + x7+ x9, data=lhs_res)
summary(quad_reg3)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 2) + x4 + x7 + x9, data = lhs_res)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6441 -0.2730 -0.1020  0.2199  1.1354
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.26219    0.07089  17.805  < 2e-16 ***
## poly(x1, 2)1 -1.37778    0.42486  -3.243  0.00139 **
## poly(x1, 2)2 -6.80255    0.42578 -15.977  < 2e-16 ***
## x4            0.82781    0.07373  11.227  < 2e-16 ***
## x7           -0.19053    0.07366  -2.587  0.01042 *
## x9           -2.04084    0.07367 -27.703  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4248 on 194 degrees of freedom
## Multiple R-squared:   0.86,  Adjusted R-squared:  0.8564
## F-statistic: 238.3 on 5 and 194 DF,  p-value: < 2.2e-16
```

```
predict.lm(quad_reg1, test)
```

```
##         1         2
## 0.8314673 1.0367868
```

```
predict.lm(quad_reg2, test)
```

```
##         1         2
## 0.8742373 1.0776445
```

```
predict.lm(quad_reg3, test)
```

```
##        1        2
## 0.845268 1.013114
```

Something is still missing…

Let's increase the polynomial degrees.

```
cub_reg1 <- lm(y ~ poly(x1,3) + poly(x4,3) + poly(x7,3) + poly(x9,3), data=lhs_res)
summary(cub_reg1)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 3) + poly(x4, 3) + poly(x7,
```

```
##     3) + poly(x9, 3), data = lhs_res)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39668 -0.11052  0.00627  0.10511  0.50264
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.56050    0.01233  45.460  < 2e-16 ***
## poly(x1, 3)1    -1.36338    0.17478  -7.801 4.21e-13 ***
## poly(x1, 3)2    -6.35847    0.31222 -20.366  < 2e-16 ***
## poly(x1, 3)3    -5.41563    0.17678 -30.635  < 2e-16 ***
## poly(x4, 3)1     4.80722    0.17521  27.436  < 2e-16 ***
## poly(x4, 3)2    -0.54486    0.29963  -1.818   0.0706 .
## poly(x4, 3)3     0.34873    0.17991   1.938   0.0541 .
## poly(x7, 3)1    -0.87745    0.17556  -4.998 1.33e-06 ***
## poly(x7, 3)2    -0.35746    0.28877  -1.238   0.2173
## poly(x7, 3)3     0.04907    0.17668   0.278   0.7815
## poly(x9, 3)1   -11.67538    0.17491 -66.749  < 2e-16 ***
## poly(x9, 3)2     0.35856    0.31236   1.148   0.2525
## poly(x9, 3)3     0.40496    0.17620   2.298   0.0227 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1744 on 187 degrees of freedom
## Multiple R-squared:  0.9773, Adjusted R-squared:  0.9758
## F-statistic: 669.7 on 12 and 187 DF,  p-value: < 2.2e-16
```

```r
cub_reg2 <- lm(y ~ poly(x1,3) + poly(x4,3) + poly(x9,3), data=lhs_res)
summary(cub_reg2)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 3) + poly(x4, 3) + poly(x9,
##     3), data = lhs_res)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44950 -0.12322 -0.00473  0.10304  0.51519
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.56050    0.01307  42.882  < 2e-16 ***
## poly(x1, 3)1    -1.37275    0.18493  -7.423 3.74e-12 ***
## poly(x1, 3)2    -6.39802    0.32032 -19.974  < 2e-16 ***
## poly(x1, 3)3    -5.44370    0.18534 -29.372  < 2e-16 ***
## poly(x4, 3)1     4.82747    0.18569  25.997  < 2e-16 ***
## poly(x4, 3)2    -0.64545    0.30578  -2.111   0.0361 *
## poly(x4, 3)3     0.25601    0.18889   1.355   0.1769
## poly(x9, 3)1   -11.68143    0.18538 -63.014  < 2e-16 ***
## poly(x9, 3)2     0.18591    0.31570   0.589   0.5566
## poly(x9, 3)3     0.37117    0.18628   1.993   0.0477 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1848 on 190 degrees of freedom
## Multiple R-squared:  0.974,  Adjusted R-squared:  0.9728
## F-statistic: 791.9 on 9 and 190 DF,  p-value: < 2.2e-16
```

```
cub_reg3 <- lm(y ~ poly(x1,3) + x4 + x7+ x9, data=lhs_res)
summary(cub_reg3)
```

```
##
## Call:
## lm.default(formula = y ~ poly(x1, 3) + x4 + x7 + x9, data = lhs_res)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37649 -0.11601  0.00838  0.09944  0.44655
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.22552    0.02971  41.256  < 2e-16 ***
## poly(x1, 3)1 -1.37763    0.17788  -7.745 5.28e-13 ***
## poly(x1, 3)2 -6.80308    0.17827 -38.163  < 2e-16 ***
## poly(x1, 3)3 -5.38411    0.17812 -30.227  < 2e-16 ***
## x4            0.83507    0.03087  27.049  < 2e-16 ***
## x7           -0.14594    0.03087  -4.727 4.39e-06 ***
## x9           -2.01937    0.03085 -65.454  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1779 on 193 degrees of freedom
## Multiple R-squared:  0.9756, Adjusted R-squared:  0.9748
## F-statistic:  1285 on 6 and 193 DF,  p-value: < 2.2e-16
```

```
predict.lm(cub_reg1, test)
```

```
##        1        2
## 1.377967 1.200843
```

```
predict.lm(cub_reg2, test)
```

```
##        1        2
## 1.400018 1.294168
```

```
predict.lm(cub_reg3, test)
```

```
##        1        2
## 1.376461 1.329212
```