Final Project

Ovie Soman, Qianli Wu, Sebastian Rivera-Chepetla

05 June, 2022

Contents

Question 1. Propose a fractional factorial design for the problem. In addition, propose an experimental design constructed using the optimal design approach.

The goal of this project is to identify the tuning parameters of random forest that affect the cross-validation accuracy.

- We have seven tuning parameters:

	Parameter	Low Level	High Level
A	ntree	100	1000
B	replace	0	1
C	mtry	2	6
D	nodesize	1	11
E	maxnodes	10	1000
F	classwt	0.5	0.9
G	cutoff	0.2	0.8

Fractional Factorial Design

Since there are 7 factors each with 2 levels, a complete factorial design 2^7 requires 128 runs. Only 7 degrees of freedom correspond to main effects, 21 correspond to two-factor interactions, and the rest of 99 are associated with three-factor and higher interactions.

Due to limitation of resources, we can only run at most 35 tests. We assume that certain high-order interactions are negligible.

Since

- The effect sparsity principle
- The projection property
- Sequential experimentation

We choose the One-Quarter Fraction of the 2^7 Design, i.e., 2^{7-2} Design.

Letting the ‘FrF2’ function recommend one design. Generally, the recommended designs are those in Table 8.14 in Chapter 8. The designs recommended by the function have the largest possible resolution. They also have the smallest aberration as defined in Section 8.4.1. Fractional factorial designs with minimum aberration are preferred because they minimize the aliasing among the factors’ effects.

```r
factors <- list(ntree = c(100, 1000),
                mtry = c(2, 6),
                replace = c(0, 1),
                nodesize = c(1,11),
                classwt = c(0.5, 0.9),
                cutoff = c(0.2, 0.8),
                maxnodes = c(10, 1000))
my.design <- FrF2(nruns = 32, nfactors = 7, randomize = F, factor.names = factors)
print(my.design)
```

```r
design.info(my.design)$catlg.entry
design.info(my.design)$aliased
```

An Experimental Design Constructed using the Optimal Design Approach.

Suggestion is to start with 32 runs

3 runs are saved for confirmation experiments or follow-up experiment.

```r
# Create a full design for 7 factors each with 2 levels
full.design <- FrF2(nruns = 128, nfactors = 7, randomize = F, factor.names = factors)
# Consider a model with 35 runs
alternative.design <- optFederov(~.^2,
                                  full.design, nTrials = 32, nRepeats = 1000)
print.data.frame(alternative.design$design)
```

Question 2. Compare the optimal design with the fractional factorial design in practical and statistical terms. For instance, what is the performance of the designs for studying the main effects of the tuning parameters only? Can they estimate all two-parameter interactions? Why or why not? How do they compare in terms of multicollinearity?

2.1: what is the performance of the designs for studying the main effects of the tuning parameters only?

```r
# Visualize the aliasing in the design.
FD.alt <- (desnum(my.design)) # Extract the design.
# Create the model matrix including main effects and two-factor interactions.
FX.alt <- model.matrix(~., data.frame(FD.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(FX.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```

2.1.1 Color Map for Main Effects of Fractional Factorial Design Since all cells other than the diagonal are white, there's no alias among all main effects and intercept for fractional factorial design. Therefore, we have the performance of fractional factorial design is good for main effects and intercept.

```r
# Visualize the aliasing in the design.
OD.alt <- alternative.design$design # Extract the design.

# Convert factors to -1 and 1
OD.alt <- sapply(OD.alt,function(x)(as.integer(x) -1.5)*2 )
OX.alt <- model.matrix(~., data.frame(OD.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(OX.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```

2.1.2 Color Map for Main Effects of D-optimal Design

2.1.3 VIF for Main Effects of D-Optimal Design

```r
# We need to include the intercept when computing the VIF.
X.alt <- model.matrix(~., data.frame(OD.alt))
# Variance-covariance matrix of 46-run design. Assuming sigma^2 = 1
var.eff.one <- diag(solve(t(X.alt)%*%X.alt))
# Set the left margin of plot be 1
par(oma=c(0,1,0,0))
#create horizontal bar chart to display each VIF value
barplot(nrow(X.alt)*var.eff.one, main = "VIF Values", horiz = TRUE, col = "cyan", las=1, cex.names=0.4,
#add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
```

From the VIF plot above, we have there's no factor has VIF greater than 5. Thus, we don't have a severe multi-collineairty or aliasing issue.

2.2 Can they estimate all two-parameter interactions?

2.2.1 Color Map for Fractional Factorial Design

```r
# Create the model matrix including main effects and two-factor interactions.
FX.alt <- model.matrix(~.^2-1, data.frame(FD.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(FX.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```

From the color map above, we get there's alias among some pairs of two-parameter interactions.

From the results of `design.info` in Question 1, we have AB=CF, AC=BF, and AF=BC, where `A=ntree`, `B=mtry`, `C=replace`, `D=nodesize`, `E=classwt`, `F=cutoff`, and `G=maxnodes`. The aliasing is also shown on the map above, as the corresponding dark cells for `mtry:classwt` with `ntree:replace`, `replace:classwt` with `ntree:mtry`, and `replace:mtry` with `ntree:classwt`.

Therefore, we have the fractional factorial design's performance is best for all main effects and two-factor interactions except these aliasing factors (AB, CF, AC, BF, AF, and BC).

2.1 Color Map for D-optimal Design

```r
OX.alt <- model.matrix(~.^2-1, data.frame(OD.alt))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(OX.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```

From the D-optimal Design above, we have most cells with light colors, which means the strength of correlation between the main effects and two-parameter interactions are weak. Therefore, we have the degree of aliasing among the main effects, between the main effects and the two-parameter interactions, and among the two-parameter interactions exist but are fairly low.

2.2.3 VIF for D-Optimal Design

```
# We need to include the intercept when computing the VIF.
X.alt <- model.matrix(~.^2, data.frame(OD.alt))
# Variance-covariance matrix of 46-run design. Assuming sigma^2 = 1
var.eff.one <- diag(solve(t(X.alt)%*%X.alt))
# Set the left margin of plot be 1
par(oma=c(0,1,0,0))
#create horizontal bar chart to display each VIF value
barplot(nrow(X.alt)*var.eff.one, main = "VIF Values", horiz = TRUE, col = "cyan", las=1, cex.names=0.4,
#add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
```

From the VIF plot above, we have there's no factor has VIF greater than 5. Thus, we don't have a severe multi-collineairty or aliasing issue.

3. Compare

- We have the fractional factorial design only suffers from multi-collinearity problem of these three pairs of aliased interactions. It will be a good design if `mtry:classwt`, `ntree:replace`, `replace:classwt`, `ntree:mtry`, `replace:mtry`, and `ntree:classwt` are not potentially important effects.
- However, there's no serious multi-collinearity problem or aliasing issue for the D-optimal design. But the variances of coefficient estimates of both main effects and two-parameter interactions are inflated trivially due to collinearity.

```r
# We compare the run size of each runs for both designs
summary(as.data.frame(my.design))
summary(alternative.design$design)
```

- On practical term, since both design require 32 runs and there's no huge difference between the proportion of two levels, we have the time it takes to run the experiments are similar for both designs.

Question 3. Recommend one experimental design between the two options in Question 1. Motivate your decision.

- We select the D-optimal Design since there's no series multi-collinearity problem or aliasing issue for all main effects and two-factor interactions.
- Also, D-optimal designs minimize the variance- covariance matrix of the coefficient's estimates and thus they are attractive to develop linear models that allow to make precise inferences. Our goal is to find the most important tuning parameters.

Question 4. Using a commercial software, the TAs and I came up with the experimental design shown in Table 2. How does your recommended design in the previous question compare with this one?

```r
table2 <- read.csv("Data/Alternative_Experimental_Design.csv")
head(table2)
```

4.1 Color Map for Table 2 Design

```
# Convert factors to -1 and 1
table2 <- sapply(table2,function(x)(as.integer(as.factor(x)) -2) )
table2[,3][table2[,3]==0] <- 1
X.alt <- model.matrix(~.^2-1, data.frame(table2[,-1]))

# Create color map on pairwise correlations.
contrast.vectors.correlations.alt <- cor(X.alt)
corrplot(contrast.vectors.correlations.alt, type = "full", addgrid.col = "gray",
         tl.col = "black", tl.srt = 90, method = "color", tl.cex=0.5)
```

- The experimental design shown in Table2 is 7 factor with 3 levels, however, the design we proposed in Question 1 has only 2 levels for each factor.
- From the color map above, we have the design in Table 2 has lighter cells for aliasing among main effects with other main effects and there's no alias between main effects and two-parameter interactions. However, since the cells at bottom right corner are darker, we have the degree of aliasing is generally greater then these for our D-optimal design for two-parameter interactions. It is reasonable since proposed design only has 22 runs, which is less then our D-optimal design with 32 runs.
- Since the proposed design is a 3-level design, it allows quadratic model, which includes x^2 as predictors. However, our 2-level D-optimal design doesn't support that. But our design has better performance on estimating two-parameter interactions.

Question 5. Collect data using your recommended design in Question 3.

```
# import function cv.rf(design, y, X)
source("CrossValidation_RandomForest/CrossValidation_RF.R")
# import dataset Cardiovascular
load("Data/cardiovascular.Rdata")
```

- We need to randomized the run order of the experiments to ensure the independence of errors.

```
# convert the design from factor to numerical values
design.numerical <- as.data.frame(sapply(alternative.design$design, function(x) as.numeric(as.character
# randomize the run order
design.randomized <- design.numerical[sample(1:nrow(design.numerical)),]
head(design.randomized)
```

```
result <- cv.rf(design.randomized, y, X)
# save a copy of result thus no need to run cv.rf again
result.copy <- result
print(result)
```

Question 6. Conduct a detailed data analysis. What are the influential tuning parameters? What is the final model that links the tuning parameters to the cross-validation accuracy? Does the final model provide a good fit to the data?

```r
# Converting numerical variables back to factors
str(result)
result[,c("ntree","mtry","replace",
          "nodesize","classwt","cutoff","maxnodes")] <- lapply(result[,c("ntree","mtry","replace",
                                                                "nodesize","classwt","cutoff",
```

```r
# Re-encoding Factors
levels(result$ntree) = c(-1,1)
levels(result$mtry) = c(-1,1)
levels(result$replace) = c(-1,1)
levels(result$nodesize) = c(-1,1)
levels(result$classwt) = c(-1,1)
levels(result$cutoff) = c(-1,1)
levels(result$maxnodes) = c(-1,1)
result <- as.data.frame(sapply(result, function(x) as.numeric(as.character(x))))
head(result)
```

A model with interaction terms with optimal design

```r
# Running Linear Model
cv.model <- lm(CV~.^2, data=result.copy)
summary(cv.model)
```

As seen from the output above, there's no significant factors except the intercept under the level of $\alpha = 0.05$. The difference between the Multiple and Adjusted R^2 values is substantial, which indicates that the high R^2 value could be due to the inclusion of several independent factors. We would thus need a reduced model.

Bayes Information Criterion (BIC)

We use BIC to avoid overfitting.

$$BIC = nln(\frac{RSS}{n}) + ln(n)k$$

where n is the number of observations, k is the number of predictors, and $ RSS $ stands for residual square sum.
- The best model has the smallest BIC value.
- For n larger than 8, log(n) > 2, and so this is a greater penalty for complexity than AIC.
- BIC favors simpler models than AIC.

```r
library(leaps)
Best_Subset <- regsubsets(CV~.^2, data = result,
                          nbest = 1, nvmax=NULL,
                          force.in = NULL, force.out=NULL,
                          method = "exhaustive",
                          really.big=F)
summary_best_subset <- summary(Best_Subset)
plot(summary_best_subset$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")
min = which.min(summary_best_subset$bic)
points(min, summary_best_subset$bic[min], col = "red", cex = 2, pch = 20)
```

We now select the model with the lowest BIC value.

```r
### Select Model with least BIC score
# We first select the best model according to BIC.
modelwith.minimum.BIC <- which.min(summary_best_subset$bic)
best.model <- summary_best_subset$which[modelwith.minimum.BIC,][-1]
# Only keep the predictors are indicated by 'TRUE' and our response 'goldDiff'
keep <- names(best.model[best.model==T])
# Print the selected factors
a <- paste(keep,sep=" ",collapse=" + ")
a
```

We now use these predictors to construct a reduced model.

```r
# construct a reduced model based on BIC criteria
cv.model.bic <- lm(CV~classwt + cutoff + maxnodes + classwt:cutoff + classwt:maxnodes, data=result)
# T-test
summary(cv.model.bic)
```

The difference between the Multiple and Adjusted R^2 values is now lower. Under $\alpha = 0.05$, all predictors, classwt, cutoff, maxnodes, the interaction between classwt and cutoff, and the interaction between classwt and maxnodes are significant factors. We now analyse the residuals.

Analysis of Residuals

```r
# Analysis of Residuals
cv.res <- cv.model.bic$residuals
cv.fit <- cv.model.bic$fitted.values
par(mfrow = c(1,3))
qqnorm(cv.res)
qqline(cv.res)
plot(cv.fit,cv.res, main = "Residuals vs Fitted Values",
     xlab = "Predicted", ylab = "Residuals")

N <- nrow(result)
plot(x = 1:N, y = cv.res, xlab = "Run Order",
     ylab = "Residuals")
```

We see that the points on the normal Q-Q Plot adhere to the qqline except at extremities. The residuals in the residuals vs fitted are evenly distributed across each fitted value. And the distribution of residuals in the residuals vs run order plots are fairly homoskedastic. Thus, we say for our final modal the assumption of constant variance, independence, and normality are satisfied.

Running Confirmation Tests

Since we can run 35 tests in total and we have already run 32 times, we have 3 runs left for confirmation experiment.

The final fitted equation is

$$\hat{y} = 0.622765 - 0.048193E + 0.028372F + 0.015964G + 0.023461EF + 0.037590EG$$

where $\hat{y}$ is the predicted response and `A=ntree`, `B=mtry`, `C=replace`, `D=nodesize`, `E=classwt`, `F=cutoff`, and `G=maxnodes`.

Using this equation, we can find the settings that maximize the porosity percentage. To this end, we use the 'optim' function in R. We first define an objective function which we want to maximize.

```r
obj_func <- function(x){
    pred.y <- 0.622765  - 0.048193 * x[5] + 0.028372 * x[6] + 0.015964*x[7] + 0.023461 *x[5]*x[6] + 0.0
    return(-1*pred.y) # Because the 'optim' function minimizes.
}
```

Now, we use the *optim* function in R.

```r
# First, use optim() to find the optimal setting according to our model
optim(par = c(-1, -1, -1, -1,-1,-1,-1), fn = obj_func, lower = -1, upper = 1, method = "L-BFGS-B")
```

```r
# Then, construct a dataframe of 7 columns, 1 row, with the optimal setting above
design.confirm <- data.frame(ntree = -1,
                mtry = -1,
                replace = -1,
                nodesize = -1,
                classwt = -1,
                cutoff = 1,
                maxnodes = -1)
head(design.confirm)
```

```r
# Then, Get the prediction interval of the optimal setting based on our model
predict(cv.model.bic, newdata = design.confirm, interval = "predict")
```

```r
# Convert the optimal setting to numerical values
design.confirm <- data.frame(ntree = rep(100,3),
                mtry = rep(2,3),
                replace = rep(0,3),
                nodesize = rep(1,3),
                classwt = rep(0.5,3),
                cutoff = rep(0.8,3),
                maxnodes = rep(10,3))
head(design.confirm)
```

```r
# Lastly, run cv.rf() three times with optimal setting (with numerical data) to confirm
confirm.result <- cv.rf(design.confirm, y, X)
confirm.result$CV
mean(confirm.result$CV)
```

We have the results of confirmation experiments are in the range of our prediction interval (0.608844, 0.7895945).
Moreover, the mean of the results (0.6992193) is close to the fitted value 0.6970846.