101B_final_project_June_2

BK, EuijunKim, Kaili

6/02/2022

Part1:

Quesetion 1: fractional design

Make the design with runs size 32 and see color map, vif, etc.

```
set.seed(1000)
frac.design <- FrF2(nruns = 32, nfactors = 7, randomize = T)
print(desnum(frac.design))
```

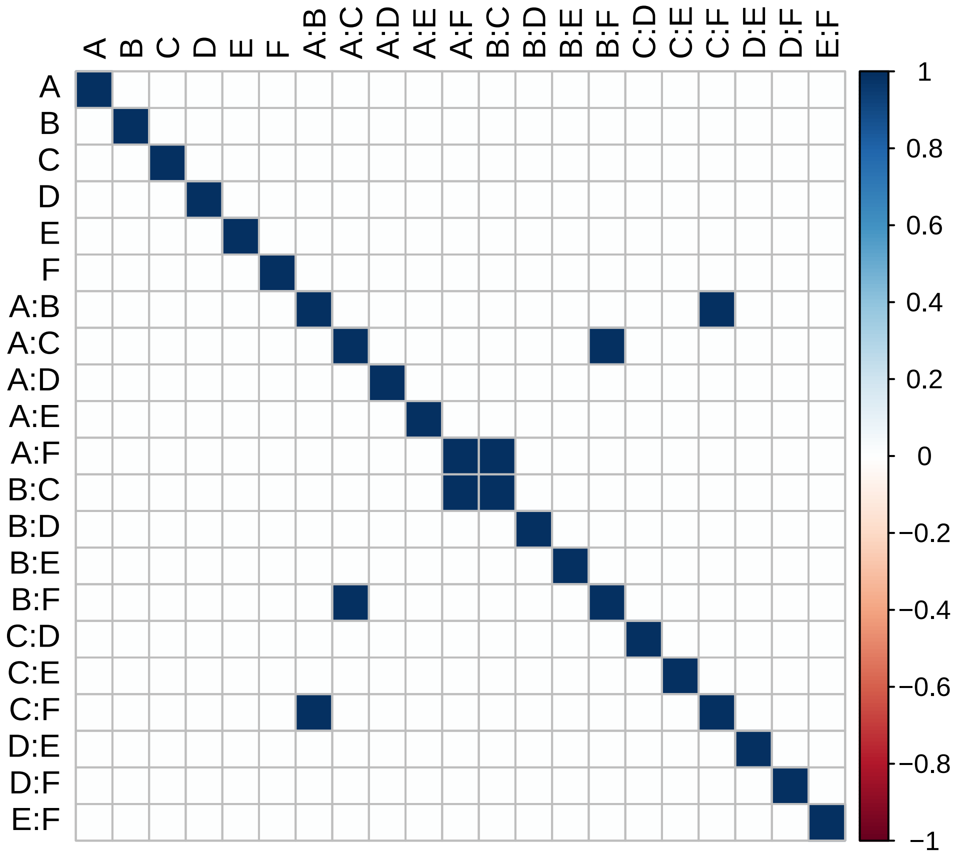
```
##     A  B  C  D  E  F  G
## 1   1  1  1  1 -1  1 -1
## 2   1  1 -1 -1 -1 -1  1
## 3  -1  1 -1  1 -1  1  1
## 4   1 -1  1 -1  1 -1  1
## 5  -1  1 -1 -1  1  1  1
## 6   1  1  1 -1  1  1 -1
## 7  -1  1 -1 -1 -1  1 -1
## 8   1 -1 -1 -1  1  1  1
## 9  -1 -1  1  1  1  1  1
## 10  1 -1  1 -1 -1 -1 -1
## 11 -1 -1  1  1 -1  1 -1
## 12 -1  1  1 -1  1 -1  1
## 13 -1 -1 -1 -1 -1 -1  1
## 14 -1 -1 -1  1 -1 -1 -1
## 15 -1 -1 -1  1  1 -1  1
## 16 -1 -1  1 -1 -1  1  1
## 17  1  1  1  1  1  1  1
## 18  1 -1  1  1 -1 -1  1
## 19  1 -1 -1  1  1  1 -1
## 20  1 -1 -1  1 -1  1  1
## 21 -1  1 -1  1  1  1 -1
## 22  1  1 -1  1  1 -1  1
## 23  1  1 -1  1 -1 -1 -1
## 24 -1  1  1 -1 -1 -1 -1
## 25  1  1  1 -1 -1  1  1
## 26 -1  1  1  1  1 -1 -1
## 27  1  1 -1 -1  1 -1 -1
## 28  1 -1  1  1  1 -1 -1
## 29 -1  1  1  1 -1 -1  1
## 30 -1 -1 -1 -1  1 -1 -1
## 31 -1 -1  1 -1  1  1 -1
## 32  1 -1 -1 -1 -1  1 -1
```

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

```
## $legend
## [1] "A=A" "B=B" "C=C" "D=D" "E=E" "F=F" "G=G"
##
## $main
## character(0)
##
## $fi2
## [1] "AB=CF" "AC=BF" "AF=BC"
```

```
D <- desnum(frac.design) # Extract the design.
# Create the model matrix including main effects and two-factor interactions.
X <- model.matrix(~(A + B + C + D + E + F)^2-1, data.frame(D))

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



```
source("CrossValidation_RF.R")
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
D.frac <- data.frame(D)
new_frac.data <- D.frac
new_frac.data[,1][new_frac.data[,1]==1] <- 1000
new_frac.data[,1][new_frac.data[,1]==-1] <- 100
new_frac.data[,2][new_frac.data[,2]==1] <- 6
new_frac.data[,2][new_frac.data[,2]==-1] <- 2
new_frac.data[,3][new_frac.data[,3]==1] <- 1
new_frac.data[,3][new_frac.data[,3]==-1] <- 0
new_frac.data[,4][new_frac.data[,4]==1] <- 11
new_frac.data[,4][new_frac.data[,4]==-1] <- 1
new_frac.data[,5][new_frac.data[,5]==1] <- 0.9
new_frac.data[,5][new_frac.data[,5]==-1] <- 0.5
new_frac.data[,6][new_frac.data[,6]==1] <- 0.8
new_frac.data[,6][new_frac.data[,6]==-1] <- 0.2
new_frac.data[,7][new_frac.data[,7]==1] <- 1000
new_frac.data[,7][new_frac.data[,7]==-1] <- 10
colnames(new_frac.data) <- c("ntree", "mtry", "replace", "nodesize", "classwt", "cutoff", "maxnodes")
new_data <- new_frac.data

load("diabetes.RData")
new.frac.design_rf <- cv.rf(new_data, y, X) # With the data actual values, we get a CV as a response va
```

```
## Collecting response on test combination  1
## Collecting response on test combination  2
## Collecting response on test combination  3
## Collecting response on test combination  4
## Collecting response on test combination  5
## Collecting response on test combination  6
## Collecting response on test combination  7
## Collecting response on test combination  8
## Collecting response on test combination  9
## Collecting response on test combination  10
## Collecting response on test combination  11
## Collecting response on test combination  12
## Collecting response on test combination  13
## Collecting response on test combination  14
## Collecting response on test combination  15
## Collecting response on test combination  16
## Collecting response on test combination  17
## Collecting response on test combination  18
## Collecting response on test combination  19
## Collecting response on test combination  20
## Collecting response on test combination  21
## Collecting response on test combination  22
## Collecting response on test combination  23
## Collecting response on test combination  24
## Collecting response on test combination  25
## Collecting response on test combination  26
## Collecting response on test combination  27
## Collecting response on test combination  28
## Collecting response on test combination  29
## Collecting response on test combination  30
## Collecting response on test combination  31
## Collecting response on test combination  32
```

```
new.frac.design <- new.frac.design_rf # matrix with actual values and CV.

coded.frac.design <- cbind(D.frac,new.frac.design$CV)
colnames(coded.frac.design) <- c("ntree", "mtry", "replace", "nodesize", "classwt", "cutoff", "maxnodes
coded.frac.design # matrix with coded values and CV.
```

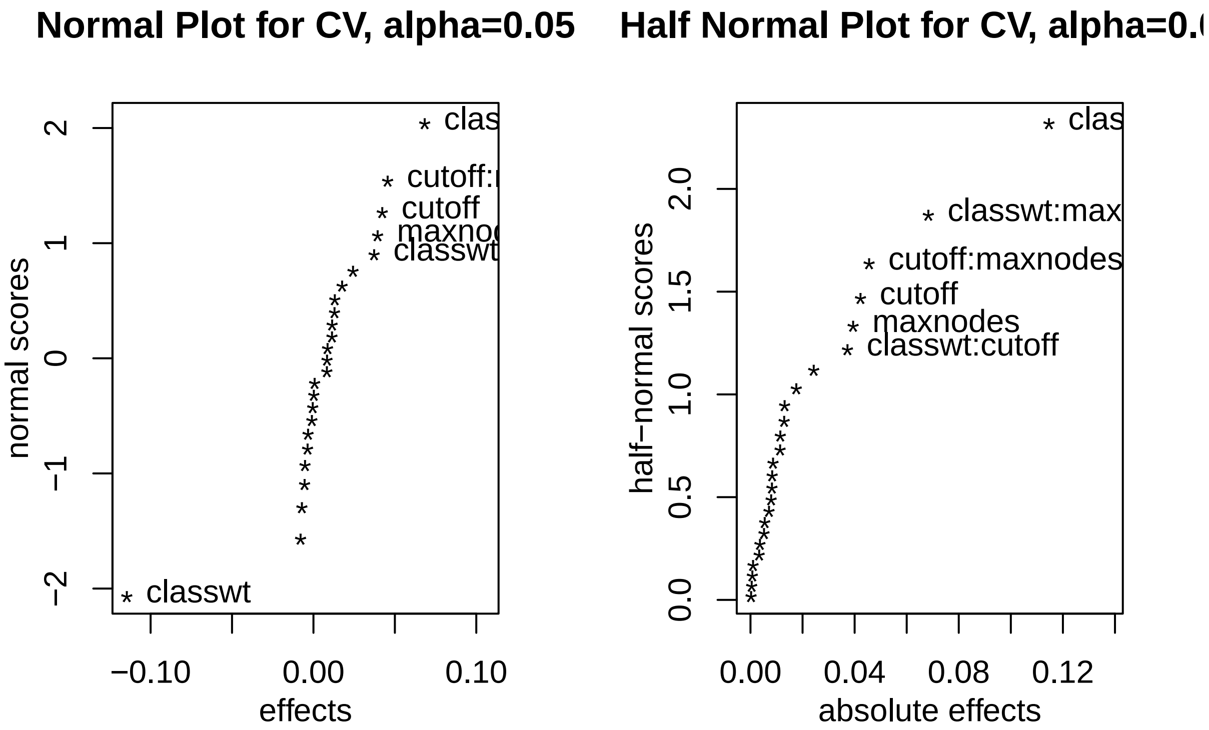
```
##    ntree mtry replace nodesize classwt cutoff maxnodes        CV
## 1      1    1       1        1      -1      1       -1 0.6936670
## 2      1    1      -1       -1      -1     -1        1 0.6256757
## 3     -1    1      -1        1      -1      1        1 0.6824084
## 4      1   -1       1       -1       1     -1        1 0.5011689
## 5     -1    1      -1       -1       1      1        1 0.6508802
## 6      1    1       1       -1       1      1       -1 0.5208853
## 7     -1    1      -1       -1      -1      1       -1 0.6955822
## 8      1   -1      -1       -1       1      1        1 0.6962893
## 9     -1   -1       1        1       1      1        1 0.6858889
## 10     1   -1       1       -1      -1     -1       -1 0.6818311
## 11    -1   -1       1        1      -1      1       -1 0.6637956
## 12    -1    1       1       -1       1     -1        1 0.6301023
## 13    -1   -1      -1       -1      -1     -1        1 0.6574491
## 14    -1   -1      -1        1      -1     -1       -1 0.6846669
## 15    -1   -1      -1        1       1     -1        1 0.5001023
## 16    -1   -1       1       -1      -1      1        1 0.6683292
## 17     1    1       1        1       1      1        1 0.7157822
## 18     1   -1       1        1      -1     -1        1 0.6614709
## 19     1   -1      -1        1       1      1       -1 0.5000578
## 20     1   -1      -1        1      -1      1        1 0.6851604
## 21    -1    1      -1        1       1      1       -1 0.5166346
## 22     1    1      -1        1       1     -1        1 0.5202358
## 23     1    1      -1        1      -1     -1       -1 0.7096264
## 24    -1    1       1       -1      -1     -1       -1 0.7075201
## 25     1    1       1       -1      -1      1        1 0.6521690
## 26    -1    1       1        1       1     -1       -1 0.5000000
## 27     1    1      -1       -1       1     -1       -1 0.5000001
## 28     1   -1       1        1       1     -1       -1 0.5000000
## 29    -1    1       1        1      -1     -1        1 0.6383911
## 30    -1   -1      -1       -1       1     -1       -1 0.5000001
## 31    -1   -1       1       -1       1      1       -1 0.5010490
## 32     1   -1      -1       -1      -1      1       -1 0.6655513
```

```
### use summary() to see significant effects
model <- lm(CV~.^2, data = coded.frac.design)
summary(model)
```

```
##
```

```
## Call:
## lm.default(formula = CV ~ .^2, data = coded.frac.design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.035267 -0.003779  0.000000  0.003779  0.035267
##
## Coefficients: (3 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.6160116  0.0059075 104.276 5.24e-11 ***
## ntree             -0.0016634  0.0059075  -0.282  0.78773
## mtry               0.0064609  0.0059075   1.094  0.31605
## replace            0.0041166  0.0059075   0.697  0.51198
## nodesize           0.0001064  0.0059075   0.018  0.98621
## classwt           -0.0573193  0.0059075  -9.703 6.88e-05 ***
## cutoff             0.0211215  0.0059075   3.575  0.01171 *
## maxnodes           0.0197074  0.0059075   3.336  0.01569 *
## ntree:mtry        -0.0035539  0.0059075  -0.602  0.56946
## ntree:replace     -0.0025930  0.0059075  -0.439  0.67608
## ntree:nodesize     0.0087954  0.0059075   1.489  0.18710
## ntree:classwt     -0.0002265  0.0059075  -0.038  0.97066
## ntree:cutoff       0.0057255  0.0059075   0.969  0.36988
## ntree:maxnodes    -0.0018116  0.0059075  -0.307  0.76947
## mtry:replace              NA         NA      NA       NA
## mtry:nodesize     -0.0004858  0.0059075  -0.082  0.93714
## mtry:classwt       0.0041618  0.0059075   0.705  0.50753
## mtry:cutoff               NA         NA      NA       NA
## mtry:maxnodes     -0.0027243  0.0059075  -0.461  0.66093
## replace:nodesize   0.0121399  0.0059075   2.055  0.08565 .
## replace:classwt    0.0065507  0.0059075   1.109  0.30994
## replace:cutoff            NA         NA      NA       NA
## replace:maxnodes   0.0043273  0.0059075   0.733  0.49147
## nodesize:classwt  -0.0039610  0.0059075  -0.671  0.52749
## nodesize:cutoff    0.0056848  0.0059075   0.962  0.37306
## nodesize:maxnodes  0.0003546  0.0059075   0.060  0.95409
## classwt:cutoff     0.0186196  0.0059075   3.152  0.01977 *
## classwt:maxnodes   0.0341566  0.0059075   5.782  0.00117 **
## cutoff:maxnodes    0.0227729  0.0059075   3.855  0.00841 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03342 on 6 degrees of freedom
## Multiple R-squared:  0.9694, Adjusted R-squared:  0.8418
## F-statistic: 7.597 on 25 and 6 DF,  p-value: 0.008985
```

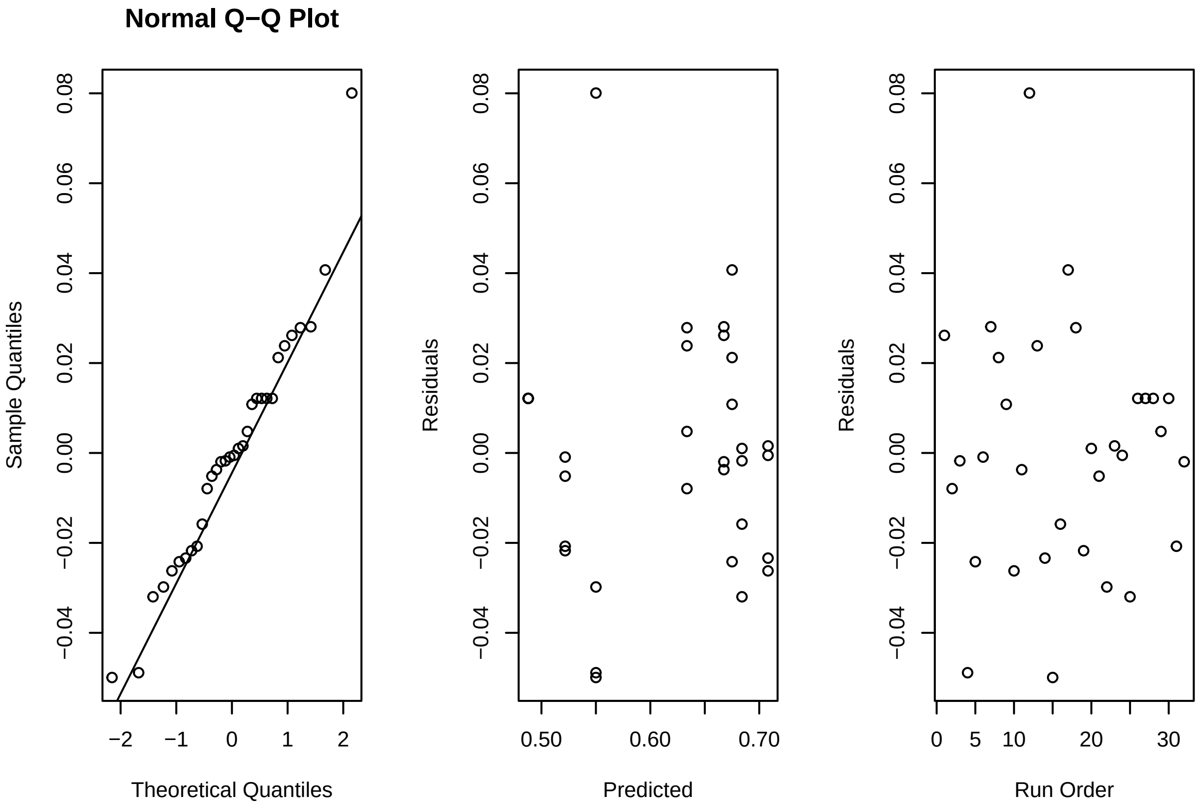
```
### we could also use DanielPlot() to check significant effets
par(mfrow = c(1,2))
DanielPlot(model, half =F, cex.fac = 1, cex.lab = 1, cex.pch = 1, cex.legend = 1)
DanielPlot(model, half =T, cex.fac = 1, cex.lab = 1, cex.pch = 1, cex.legend = 1)
```



```
### based on the result above (summary() and DanielPlot), we make a model.reduced.
model.reduced <- lm(CV ~ classwt + cutoff + maxnodes + classwt:maxnodes + cutoff:maxnodes +
      classwt:cutoff, data = coded.frac.design
  )
summary(model.reduced)
```

```
##
## Call:
## lm.default(formula = CV ~ classwt + cutoff + maxnodes + classwt:maxnodes +
##     cutoff:maxnodes + classwt:cutoff, data = coded.frac.design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.049940 -0.020995 -0.000721  0.012140  0.080060
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.616012   0.005238 117.613  < 2e-16 ***
## classwt          -0.057319   0.005238 -10.944 5.04e-11 ***
## cutoff            0.021122   0.005238   4.033 0.000456 ***
## maxnodes          0.019707   0.005238   3.763 0.000909 ***
## classwt:maxnodes  0.034157   0.005238   6.521 7.86e-07 ***
## cutoff:maxnodes   0.022773   0.005238   4.348 0.000202 ***
## classwt:cutoff    0.018620   0.005238   3.555 0.001538 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02963 on 25 degrees of freedom
## Multiple R-squared:  0.8997, Adjusted R-squared:  0.8756
## F-statistic: 37.38 on 6 and 25 DF,  p-value: 2.643e-11
```

```
# q-q and residual plots
yield.resid <- residuals(model.reduced)
pred.yield <- fitted(model.reduced)
par(mfrow = c(1,3))
qqnorm(yield.resid); qqline(yield.resid)
plot(x = pred.yield, y = yield.resid,
     xlab = "Predicted", ylab = "Residuals")
plot(x = 1:32, y = yield.resid, xlab = "Run Order",
     ylab = "Residuals")
```



```
# In residual plot, we can find one outlier(let's say it's okay), and the variance plots look okay.
```

VIF for model.

```
X.frac <- model.matrix(~(classwt + cutoff + maxnodes + classwt:maxnodes + cutoff:maxnodes +
      classwt:cutoff)^2-1, data.frame(new_data)) # matrix with main effects and
var.eff.one <- diag(solve(t(X.frac)%*%X.frac))
```

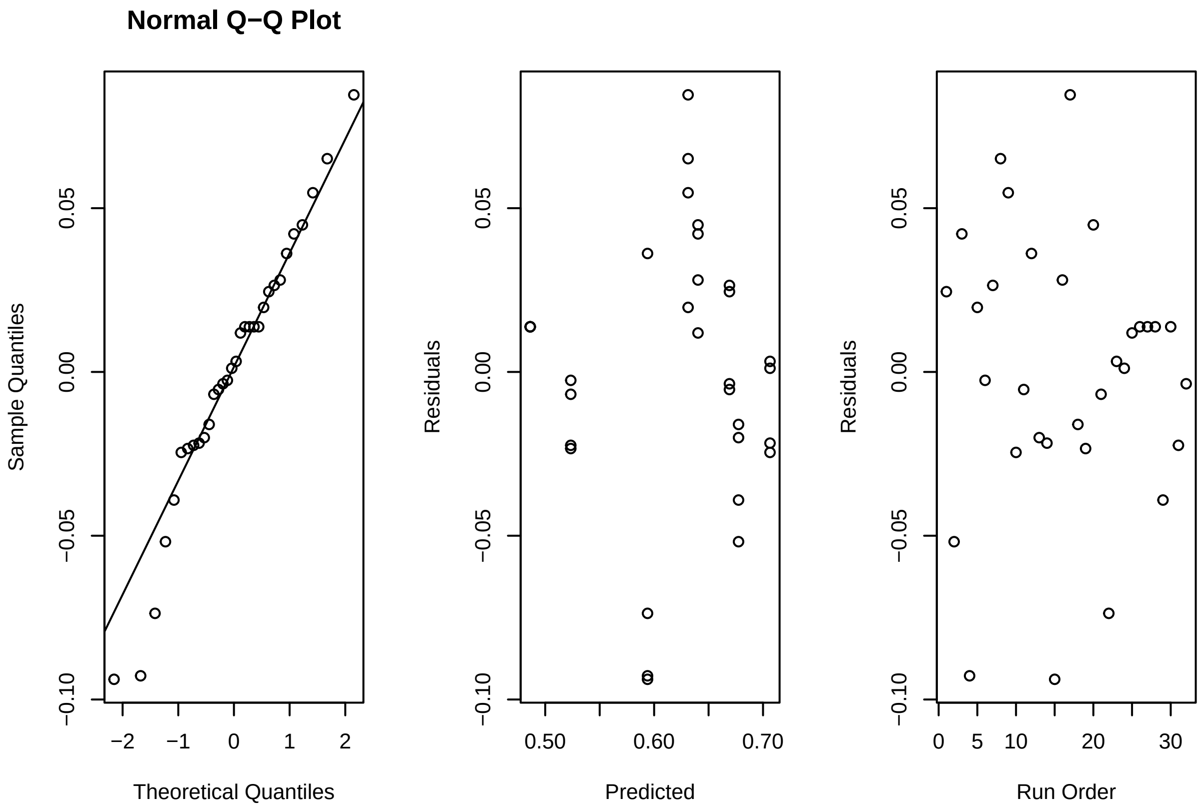
```r
results.frac <- data.frame('Var.32run' = var.eff.one, 'VIF.32run' = nrow(X.frac)*var.eff.one)
print.data.frame(results.frac)
```

```
##                            Var.32run    VIF.32run
## classwt                 4.545834e-01 1.454667e+01
## cutoff                  2.485364e+00 7.953165e+01
## maxnodes                3.128159e-06 1.001011e-04
## classwt:cutoff          5.672462e+00 1.815188e+02
## classwt:maxnodes        6.365815e-06 2.037061e-04
## cutoff:maxnodes         1.173528e-05 3.755290e-04
## classwt:cutoff:maxnodes 2.314470e-05 7.406303e-04
```

```r
#### VIF shows that cutoff seems to be problematic
### So, let's get rid of cutoff:maxnodes from the model and make a new.model.reduced.
### And then create q-q plot and residual plots.
new.model.reduced <- lm(CV ~ classwt + maxnodes + classwt:maxnodes +
      classwt:cutoff, data = coded.frac.design
  )
summary(new.model.reduced)
```

```
## 
## Call:
## lm.default(formula = CV ~ classwt + maxnodes + classwt:maxnodes + 
##     classwt:cutoff, data = coded.frac.design)
## 
## Residuals:
##       Min        1Q    Median        3Q       Max 
## -0.093834 -0.021899  0.002174  0.024985  0.084606 
## 
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)       0.616012   0.007819  78.788  < 2e-16 ***
## classwt          -0.057319   0.007819  -7.331 6.93e-08 ***
## maxnodes          0.019707   0.007819   2.521 0.017932 *  
## classwt:maxnodes  0.034157   0.007819   4.369 0.000166 ***
## classwt:cutoff    0.018620   0.007819   2.381 0.024557 *  
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
## 
## Residual standard error: 0.04423 on 27 degrees of freedom
## Multiple R-squared:  0.7586, Adjusted R-squared:  0.7229 
## F-statistic: 21.21 on 4 and 27 DF,  p-value: 5.217e-08
```

```r
yield.resid <- residuals(new.model.reduced)
pred.yield <- fitted(new.model.reduced)
par(mfrow = c(1,3))
qqnorm(yield.resid); qqline(yield.resid)
plot(x = pred.yield, y = yield.resid,
     xlab = "Predicted", ylab = "Residuals")
plot(x = 1:32, y = yield.resid, xlab = "Run Order",
     ylab = "Residuals")
```



```
### Let's see new VIF again
new.X.frac <- model.matrix(~(classwt + maxnodes + classwt:maxnodes  +
      classwt:cutoff)^2-1, data.frame(new_data)) # matrix with main effects and
var.eff.one.tw0 <- diag(solve(t(new.X.frac)%*%new.X.frac))

results.frac <- data.frame('Var.32run' = var.eff.one.tw0, 'VIF.32run' = nrow(new.X.frac)*var.eff.one.tw
print.data.frame(results.frac)
```

```
##                             Var.32run    VIF.32run
## classwt                  4.545834e-01 1.454667e+01
## maxnodes                 8.280422e-07 2.649735e-05
## classwt:maxnodes         2.353507e-06 7.531222e-05
## classwt:cutoff           1.337010e+00 4.278432e+01
## classwt:maxnodes:cutoff 2.673753e-06 8.556009e-05
```

```
### As a result from new.reduced.model, summary() and VIF do not give a better result.
### So, for now, I will just go with the first reduced.model.
```

Question1: optimal design

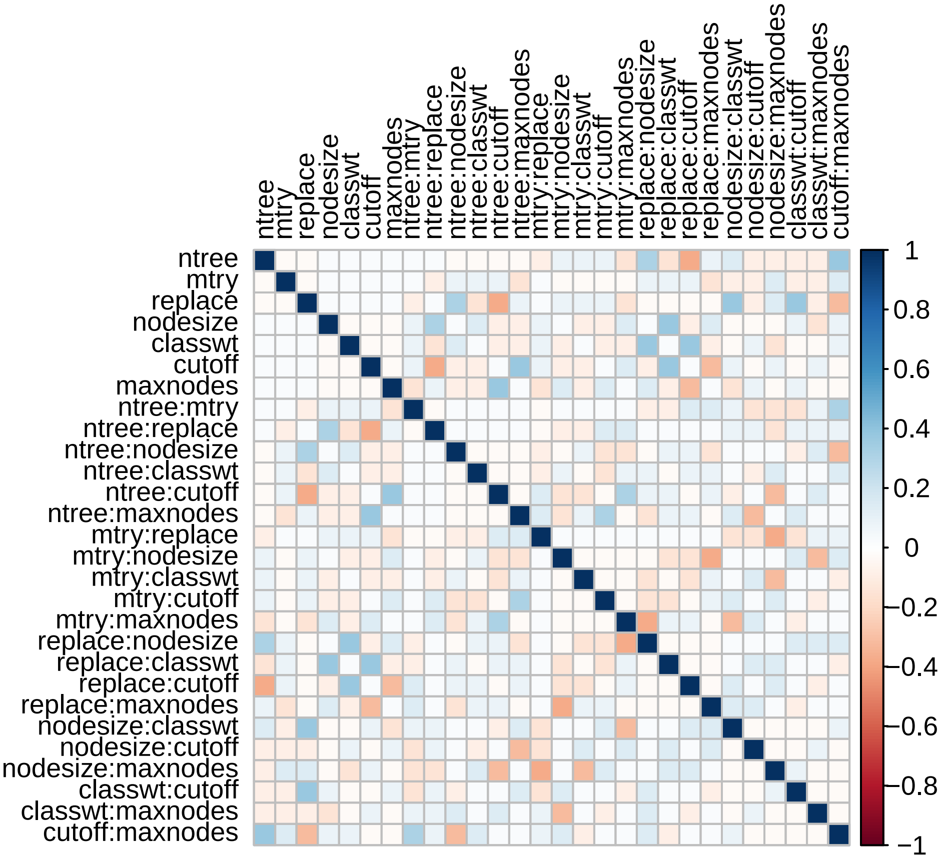
Make the design with run size 35 and see color plot

```
set.seed(1000)
my.design <- gen.factorial(levels=2, nVars = 7,
                              varNames = c("ntree", "mtry", "replace", "nodesize", "classwt","cutoff",

opt.design <- optFederov(~.,my.design, nTrials = 35, nRepeats = 1000)

D.opt <- opt.design$design # Extract the design.
# We can visualize the aliasing in this design using a color map on correlations.
# Create the model matrix including main effects and two-factor interactions.
X.opt <- model.matrix(~(ntree + mtry + replace + nodesize + classwt + cutoff + maxnodes)^2-1, data.fram

# Create color map on pairwise correlations.
contrast.vectors.correlations.opt <- cor(X.opt)
corrplot(
  contrast.vectors.correlations.opt,
  type = "full",
  addgrid.col = "gray",
  tl.col = "black",
  tl.srt = 90,
  method = "color",
  tl.cex = 0.8
)
```



VIF for model.

```
var.eff.one1 <- diag(solve(t(X.opt)%*%X.opt))

results.opt <- data.frame('Var.35run' = var.eff.one1, 'VIF.35run' = nrow(X.opt)*var.eff.one1)
print.data.frame(results.opt)
```

```
##                    Var.35run VIF.35run
## ntree             0.08922772  3.122970
## mtry              0.04061037  1.421363
## replace           0.36330573 12.715701
## nodesize          0.07035283  2.462349
## classwt           0.08009557  2.803345
## cutoff            0.15530419  5.435647
## maxnodes          0.16493098  5.772584
## ntree:mtry        0.08742056  3.059720
## ntree:replace     0.10260262  3.591092
## ntree:nodesize    0.10456142  3.659650
## ntree:classwt     0.03787607  1.325662
## ntree:cutoff      0.33133136 11.596597
## ntree:maxnodes    0.06801427  2.380499
## mtry:replace      0.19896379  6.963733
## mtry:nodesize     0.13366595  4.678308
## mtry:classwt      0.05537739  1.938209
## mtry:cutoff       0.05071482  1.775019
## mtry:maxnodes     0.18397234  6.439032
## replace:nodesize  0.07684919  2.689722
## replace:classwt   0.07131243  2.495935
## replace:cutoff    0.08083831  2.829341
## replace:maxnodes  0.11435907  4.002568
## nodesize:classwt  0.25098104  8.784336
## nodesize:cutoff   0.06152198  2.153269
## nodesize:maxnodes 0.09795519  3.428432
## classwt:cutoff    0.19642274  6.874796
## classwt:maxnodes  0.05655215  1.979325
## cutoff:maxnodes   0.10004042  3.501415
```

```
#### VIF looks okay
```

Use cv.rf function to have a response variable

```
new_data1 <- D.opt
new_data1[,1][new_data1[,1]==1] <- 1000
new_data1[,1][new_data1[,1]==-1] <- 100
new_data1[,2][new_data1[,2]==1] <- 6
new_data1[,2][new_data1[,2]==-1] <- 2
new_data1[,3][new_data1[,3]==1] <- 1
new_data1[,3][new_data1[,3]==-1] <- 0
new_data1[,4][new_data1[,4]==1] <- 11
new_data1[,4][new_data1[,4]==-1] <- 1
```

```r
new_data1[,5][new_data1[,5]==1] <- 0.9
new_data1[,5][new_data1[,5]==-1] <- 0.5
new_data1[,6][new_data1[,6]==1] <- 0.8
new_data1[,6][new_data1[,6]==-1] <- 0.2
new_data1[,7][new_data1[,7]==1] <- 1000
new_data1[,7][new_data1[,7]==-1] <- 10
print(new_data1) # data with actual vales
```

```
##     ntree mtry replace nodesize classwt cutoff maxnodes
## 3     100    6       0        1     0.5    0.2       10
## 6    1000    2       1        1     0.5    0.2       10
## 9     100    2       0       11     0.5    0.2       10
## 16   1000    6       1       11     0.5    0.2       10
## 18   1000    2       0        1     0.9    0.2       10
## 20   1000    6       0        1     0.9    0.2       10
## 25    100    2       0       11     0.9    0.2       10
## 28   1000    6       0       11     0.9    0.2       10
## 32   1000    6       1       11     0.9    0.2       10
## 36   1000    6       0        1     0.5    0.8       10
## 37    100    2       1        1     0.5    0.8       10
## 39    100    6       1        1     0.5    0.8       10
## 41    100    2       0       11     0.5    0.8       10
## 42   1000    2       0       11     0.5    0.8       10
## 53    100    2       1        1     0.9    0.8       10
## 55    100    6       1        1     0.9    0.8       10
## 62   1000    2       1       11     0.9    0.8       10
## 63    100    6       1       11     0.9    0.8       10
## 69    100    2       1        1     0.5    0.2     1000
## 70   1000    2       1        1     0.5    0.2     1000
## 75    100    6       0       11     0.5    0.2     1000
## 78   1000    2       1       11     0.5    0.2     1000
## 79    100    6       1       11     0.5    0.2     1000
## 81    100    2       0        1     0.9    0.2     1000
## 83    100    6       0        1     0.9    0.2     1000
## 87    100    6       1        1     0.9    0.2     1000
## 94   1000    2       1       11     0.9    0.2     1000
## 98   1000    2       0        1     0.5    0.8     1000
## 104  1000    6       1        1     0.5    0.8     1000
## 107   100    6       0       11     0.5    0.8     1000
## 108  1000    6       0       11     0.5    0.8     1000
## 114  1000    2       0        1     0.9    0.8     1000
## 116  1000    6       0        1     0.9    0.8     1000
## 125   100    2       1       11     0.9    0.8     1000
## 128  1000    6       1       11     0.9    0.8     1000
```

```r
load("diabetes.RData")
new.opt.design.rf <- cv.rf(new_data1, y, X) # With the data acutal values, we get a CV as a response va
```

```
## Collecting response on test combination  1
## Collecting response on test combination  2
## Collecting response on test combination  3
## Collecting response on test combination  4
## Collecting response on test combination  5
## Collecting response on test combination  6
## Collecting response on test combination  7
## Collecting response on test combination  8
## Collecting response on test combination  9
## Collecting response on test combination  10
## Collecting response on test combination  11
## Collecting response on test combination  12
## Collecting response on test combination  13
## Collecting response on test combination  14
## Collecting response on test combination  15
## Collecting response on test combination  16
## Collecting response on test combination  17
## Collecting response on test combination  18
## Collecting response on test combination  19
## Collecting response on test combination  20
## Collecting response on test combination  21
## Collecting response on test combination  22
## Collecting response on test combination  23
## Collecting response on test combination  24
## Collecting response on test combination  25
## Collecting response on test combination  26
## Collecting response on test combination  27
## Collecting response on test combination  28
## Collecting response on test combination  29
## Collecting response on test combination  30
## Collecting response on test combination  31
## Collecting response on test combination  32
## Collecting response on test combination  33
## Collecting response on test combination  34
## Collecting response on test combination  35
```

```r
new.opt.design <- new.opt.design.rf # matrix with actual values and CV.
# new.opt.design

extraction.data.coded <- cbind(D.opt,new.opt.design$CV)
colnames(extraction.data.coded)[8] <- "CV"
# extraction.data.coded # data with coded values

##### Here, we've got 2 types of data.set ( 1: with actual values 2: with coded.values)
# new.opt.design # with actual values
# extraction.data.coded # with coded values
```

```r
coded.model <- lm(CV~., data = extraction.data.coded)
summary(coded.model) # Here, we now want to find significant effects
```

```
##
## Call:
## lm.default(formula = CV ~ ., data = extraction.data.coded)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.07394 -0.04064 -0.01566  0.04080  0.11002
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.6187998  0.0096143  64.363  < 2e-16 ***
## ntree        0.0008358  0.0096143   0.087   0.9314
## mtry         0.0079794  0.0096143   0.830   0.4138
## replace     -0.0026431  0.0096143  -0.275   0.7855
## nodesize    -0.0044734  0.0096143  -0.465   0.6455
## classwt     -0.0547306  0.0096143  -5.693 4.77e-06 ***
## cutoff       0.0177435  0.0096143   1.846   0.0760 .
## maxnodes     0.0240051  0.0096143   2.497   0.0189 *
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.05668 on 27 degrees of freedom
## Multiple R-squared:  0.6218, Adjusted R-squared:  0.5238
## F-statistic: 6.342 on 7 and 27 DF,  p-value: 0.0001839
```

```
coded.model.int <- lm(CV~.^2, data = extraction.data.coded)
summary(coded.model.int) # Here, we now want to find significant effects
```

```
##
## Call:
## lm.default(formula = CV ~ .^2, data = extraction.data.coded)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.041591 -0.020273  0.002848  0.015621  0.028378
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.6223894  0.0091263  68.198 6.69e-10 ***
## ntree             -0.0052847  0.0145850  -0.362  0.72952
## mtry               0.0115837  0.0097774   1.185  0.28092
## replace           -0.0090624  0.0311111  -0.291  0.78064
## nodesize          -0.0006818  0.0129063  -0.053  0.95959
## classwt           -0.0510300  0.0137362  -3.715  0.00991 **
## cutoff             0.0234086  0.0192612   1.215  0.26989
## maxnodes           0.0248004  0.0205536   1.207  0.27299
## ntree:mtry        -0.0050591  0.0149293  -0.339  0.74625
## ntree:replace      0.0005055  0.0155612   0.032  0.97514
## ntree:nodesize     0.0056171  0.0159766   0.352  0.73717
## ntree:classwt     -0.0055235  0.0094612  -0.584  0.58062
## ntree:cutoff      -0.0006514  0.0288473  -0.023  0.98272
## ntree:maxnodes    -0.0027048  0.0127550  -0.212  0.83908
## mtry:replace       0.0049627  0.0232095   0.214  0.83777
## mtry:nodesize     -0.0002519  0.0186627  -0.013  0.98967
## mtry:classwt       0.0077227  0.0114717   0.673  0.52589
## mtry:cutoff       -0.0086406  0.0111727  -0.773  0.46867
## mtry:maxnodes      0.0025891  0.0211858   0.122  0.90672
## replace:nodesize  -0.0015711  0.0135589  -0.116  0.91153
## replace:classwt   -0.0091661  0.0130129  -0.704  0.50760
## replace:cutoff    -0.0059641  0.0141653  -0.421  0.68840
## replace:maxnodes   0.0085898  0.0168493   0.510  0.62839
## nodesize:classwt   0.0074111  0.0259237   0.286  0.78458
## nodesize:cutoff    0.0021030  0.0121429   0.173  0.86820
## nodesize:maxnodes  0.0061127  0.0156883   0.390  0.71026
## classwt:cutoff     0.0230303  0.0228151   1.009  0.35173
## classwt:maxnodes   0.0341955  0.0117437   2.912  0.02692 *
## cutoff:maxnodes    0.0195318  0.0157752   1.238  0.26191
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.04849 on 6 degrees of freedom
## Multiple R-squared:  0.9385, Adjusted R-squared:  0.6514
## F-statistic: 3.269 on 28 and 6 DF,  p-value: 0.07143
```

```
alias(coded.model.int)
```

```
## Model :
## CV ~ (ntree + mtry + replace + nodesize + classwt + cutoff +
##     maxnodes)^2
```

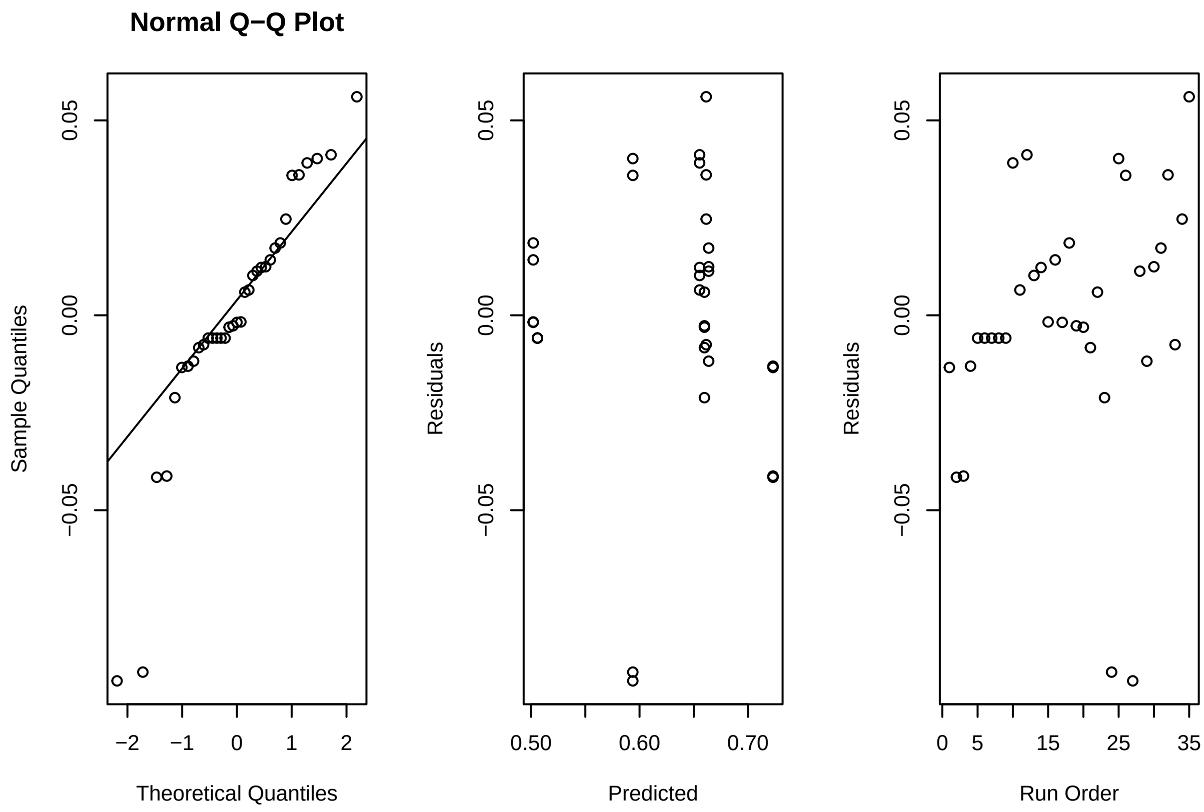
```
#### Here, Daniel plot doesn't work, and summary() gives me no info. So, I'm trying to get rid of some
### replace and ntree:cutoff are giving hihg VIF which we saw from the table above.
coded.model.reduced <- lm(CV~.^2 - replace - ntree:cutoff - nodesize:classwt, data = extraction.data.co
summary(coded.model.reduced)
```

```
##
## Call:
## lm.default(formula = CV ~ .^2 - replace - ntree:cutoff - nodesize:classwt,
##     data = extraction.data.coded)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.038524 -0.017245  0.001948  0.016683  0.031684
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        6.211e-01  7.098e-03  87.497 1.69e-14 ***
## ntree             -6.066e-03  1.137e-02  -0.534 0.606545
## mtry               1.133e-02  7.925e-03   1.430 0.186634
## nodesize          -8.473e-04  9.091e-03  -0.093 0.927785
## classwt           -5.316e-02  9.640e-03  -5.515 0.000373 ***
## cutoff             2.427e-02  1.307e-02   1.857 0.096238 .
## maxnodes           2.278e-02  8.839e-03   2.577 0.029829 *
## ntree:mtry        -7.443e-03  9.543e-03  -0.780 0.455447
## ntree:replace      1.879e-05  1.063e-02   0.002 0.998628
## ntree:nodesize     3.485e-03  8.933e-03   0.390 0.705503
## ntree:classwt     -4.611e-03  7.716e-03  -0.598 0.564855
## ntree:maxnodes    -1.949e-03  1.008e-02  -0.193 0.850944
## mtry:replace      -6.899e-04  9.718e-03  -0.071 0.944957
## mtry:nodesize      2.595e-03  1.054e-02   0.246 0.811071
## mtry:classwt       7.224e-03  8.286e-03   0.872 0.405959
## mtry:cutoff       -6.749e-03  8.709e-03  -0.775 0.458259
## mtry:maxnodes      1.079e-03  8.602e-03   0.125 0.902951
## replace:nodesize   3.623e-04  1.053e-02   0.034 0.973305
## replace:classwt   -6.758e-03  9.476e-03  -0.713 0.493821
## replace:cutoff    -2.464e-03  1.035e-02  -0.238 0.817231
## replace:maxnodes   1.105e-02  1.185e-02   0.933 0.375143
## nodesize:cutoff    1.095e-03  9.777e-03   0.112 0.913312
## nodesize:maxnodes  8.395e-04  1.000e-02   0.084 0.934932
## classwt:cutoff     1.780e-02  7.783e-03   2.287 0.047981 *
## classwt:maxnodes   3.619e-02  8.609e-03   4.204 0.002293 **
## cutoff:maxnodes    2.293e-02  9.889e-03   2.319 0.045576 *
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.04054 on 9 degrees of freedom
## Multiple R-squared:  0.9355, Adjusted R-squared:  0.7564
## F-statistic: 5.223 on 25 and 9 DF,  p-value: 0.006921
```

```
### summary(coded.model.reduced) gives us some significant effects
coded.model.reduced.two <- lm(CV~(classwt + maxnodes + classwt:cutoff + classwt:maxnodes + cutoff:maxno
                          data = extraction.data.coded)
summary(coded.model.reduced.two)
```

```
##
## Call:
## lm.default(formula = CV ~ (classwt + maxnodes + classwt:cutoff +
##     classwt:maxnodes + cutoff:maxnodes), data = extraction.data.coded)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.093783 -0.007918 -0.001815  0.015728  0.056058
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.620642   0.005842 106.246  < 2e-16 ***
## classwt          -0.054878   0.005859  -9.366 2.85e-10 ***
## maxnodes          0.024075   0.005859   4.109 0.000297 ***
## classwt:cutoff    0.015943   0.005859   2.721 0.010887 *
## classwt:maxnodes  0.037794   0.005842   6.470 4.43e-07 ***
## maxnodes:cutoff   0.017898   0.005859   3.055 0.004795 **
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.03449 on 29 degrees of freedom
## Multiple R-squared:  0.8496, Adjusted R-squared:  0.8237
## F-statistic: 32.77 on 5 and 29 DF,  p-value: 4.386e-11
```

```
# q-q and residual plots
yield.resid <- residuals(coded.model.reduced.two)
pred.yield <- fitted(coded.model.reduced.two)
par(mfrow = c(1,3))
qqnorm(yield.resid); qqline(yield.resid)
plot(x = pred.yield, y = yield.resid,
     xlab = "Predicted", ylab = "Residuals")
plot(x = 1:35, y = yield.resid, xlab = "Run Order",
     ylab = "Residuals")
```



```
### Here, we have problem. The plots look not okay actually..
```

TA's

```
source("CrossValidation_RF.R")
TAdesign <- data.frame(c(100,550,1000,1000,1000,
                         100,1000,100,100,100,
                         100,1000,100,550,100,
                         1000,1000,1000,100,1000,
                         550,550)
,c(2,2,4,6,6,
   2,2,2,6,6,
   4,2,6,4,6,
   6,2,6,2,2,
   4,6)
,c(1,0,1,1,0,
   0,0,0,1,0,
   0,1,1,0,0,
   0,0,1,1,1,
   1,1)
,c(11,1,1,1,1,
   1,6,11,1,1,
   11,11,6,6,11,
   11,11,11,1,1,
   6,11)
,c(0.5,0.5,0.5,0.5,0.9,
   0.5,0.9,0.9,0.7,0.9,
   0.9,0.5,0.5,0.7,0.5,
   0.5,0.7,0.9,0.9,0.9,
   0.7,0.9)
,c(0.8,0.2,0.2,0.8,0.2,
   0.8,0.8,0.2,0.8,0.5,
   0.8,0.5,0.2,0.5,0.2,
   0.8,0.2,0.2,0.2,0.8,
   0.5,0.8)
,c(10,10,10,1000,1000,
   1000,10,10,10,10,
   1000,1000,1000,505,505,
   10,1000,10,1000,505,
   505,1000))
colnames(TAdesign) <- c("ntree", "mtry", "replace",  "nodesize","classwt", "cutoff","maxnodes")
print(TAdesign, row.names = FALSE)
```

```
##  ntree mtry replace nodesize classwt cutoff maxnodes
##    100    2       1       11     0.5    0.8       10
##    550    2       0        1     0.5    0.2       10
##   1000    4       1        1     0.5    0.2       10
##   1000    6       1        1     0.5    0.8     1000
##   1000    6       0        1     0.9    0.2     1000
##    100    2       0        1     0.5    0.8     1000
##   1000    2       0        6     0.9    0.8       10
##    100    2       0       11     0.9    0.2       10
##    100    6       1        1     0.7    0.8       10
##    100    6       0        1     0.9    0.5       10
##    100    4       0       11     0.9    0.8     1000
##   1000    2       1       11     0.5    0.5     1000
##    100    6       1        6     0.5    0.2     1000
##    550    4       0        6     0.7    0.5      505
##    100    6       0       11     0.5    0.2      505
##   1000    6       0       11     0.5    0.8       10
##   1000    2       0       11     0.7    0.2     1000
##   1000    6       1       11     0.9    0.2       10
##    100    2       1        1     0.9    0.2     1000
##   1000    2       1        1     0.9    0.8      505
##    550    4       1        6     0.7    0.5      505
##    550    6       1       11     0.9    0.8     1000
```

```
load("diabetes.RData")
TAresults <- cv.rf(TAdesign, y, X)
```

```
## Collecting response on test combination  1
## Collecting response on test combination  2
## Collecting response on test combination  3
## Collecting response on test combination  4
## Collecting response on test combination  5
## Collecting response on test combination  6
## Collecting response on test combination  7
## Collecting response on test combination  8
## Collecting response on test combination  9
## Collecting response on test combination  10
## Collecting response on test combination  11
## Collecting response on test combination  12
## Collecting response on test combination  13
## Collecting response on test combination  14
## Collecting response on test combination  15
## Collecting response on test combination  16
## Collecting response on test combination  17
## Collecting response on test combination  18
## Collecting response on test combination  19
## Collecting response on test combination  20
## Collecting response on test combination  21
## Collecting response on test combination  22
```

```
# TAresults # matrix with CV as a respons variable

TA.coded.data <- TAresults
TA.coded.data[,1][TA.coded.data[,1]==100] <- -1
TA.coded.data[,1][TA.coded.data[,1]==550] <- 0
TA.coded.data[,1][TA.coded.data[,1]==1000] <- 1

TA.coded.data[,2][TA.coded.data[,2]==2] <- -1
TA.coded.data[,2][TA.coded.data[,2]==4] <- 0
TA.coded.data[,2][TA.coded.data[,2]==6] <- 1

TA.coded.data[,3][TA.coded.data[,3]==0] <- -1
TA.coded.data[,3][TA.coded.data[,3]==1] <- 1

TA.coded.data[,4][TA.coded.data[,4]==1] <- -1
TA.coded.data[,4][TA.coded.data[,4]==6] <- 0
TA.coded.data[,4][TA.coded.data[,4]==11] <- 1

TA.coded.data[,5][TA.coded.data[,5]==0.5] <- -1
TA.coded.data[,5][TA.coded.data[,5]==0.7] <- 0
TA.coded.data[,5][TA.coded.data[,5]==0.9] <- 1

TA.coded.data[,6][TA.coded.data[,6]==0.2] <- -1
TA.coded.data[,6][TA.coded.data[,6]==0.5] <- 0
TA.coded.data[,6][TA.coded.data[,6]==0.8] <- 1

TA.coded.data[,7][TA.coded.data[,7]==10] <- -1
TA.coded.data[,7][TA.coded.data[,7]==505] <- 0
TA.coded.data[,7][TA.coded.data[,7]==1000] <- 1


print(TA.coded.data) # data with coded vales
```

```
##    ntree mtry replace nodesize classwt cutoff maxnodes        CV
## 1     -1   -1       1        1      -1      1       -1 0.6635911
## 2      0   -1      -1       -1      -1     -1       -1 0.6820796
## 3      1    0       1       -1      -1     -1       -1 0.7031909
## 4      1    1       1       -1      -1      1        1 0.6509424
## 5      1    1      -1       -1       1     -1        1 0.6349065
## 6     -1   -1      -1       -1      -1      1        1 0.6743822
## 7      1   -1      -1        0       1      1       -1 0.5002134
## 8     -1   -1      -1        1       1     -1       -1 0.5000000
## 9     -1    1       1       -1       0      1       -1 0.7222842
## 10    -1    1      -1       -1       1      0       -1 0.5004667
## 11    -1    0      -1        1       1      1        1 0.7128980
## 12     1   -1       1        1      -1      0        1 0.7394889
## 13    -1    1       1        0      -1     -1        1 0.6258453
## 14     0    0      -1        0       0      0        0 0.7083643
## 15    -1    1      -1        1      -1     -1        0 0.6544887
## 16     1    1      -1        1      -1      1       -1 0.6939424
## 17     1   -1      -1        1       0     -1        1 0.5447868
## 18     1    1       1        1       1     -1       -1 0.5000001
## 19    -1   -1       1       -1       1     -1        1 0.5022178
## 20     1   -1       1       -1       1      1        0 0.7002000
## 21     0    0       1        0       0      0        0 0.7176668
## 22     0    1       1        1       1      1        1 0.7160623
```

```
# # plots
# yield.resid <- residuals(TA.model)
# pred.yield <- fitted(TA.model)
# par(mfrow = c(1,3))
# qqnorm(yield.resid); qqline(yield.resid)
# plot(x = pred.yield, y = yield.resid,
#      xlab = "Predicted", ylab = "Residuals")
# plot(x = 1:22, y = yield.resid, xlab = "Run Order",
#      ylab = "Residuals")
```

```
# # Create color map on pairwise correlations.
# ta.model_new <- lm(CV~ntree * mtry * replace * nodesize * classwt * cutoff * maxnodes, data = TA.code
# summary(ta.model_new)
#
# X.ta <- model.matrix(~(ntree + mtry + replace + nodesize + classwt + cutoff + maxnodes)^2-1, TA.coded
# contrast.vectors.correlations <- cor(X.ta)
# corrplot(contrast.vectors.correlations, type = "full",
#          tl.col = "black", tl.srt = 90, method = "color",
#          addgrid.col = "gray")
#
# #
#
#
# # VIF
# var.eff.one.ta <- diag(solve(t(X.ta)%*%X.ta))
#
# results.ta <- data.frame('Var.22run' = var.eff.one.ta, 'VIF.22run' = nrow(X.ta)*var.eff.one.ta)
# print.data.frame(results.ta)
```

```
# effects <- 2*(coef(ta.model_new)[-1])
# qqnorm(effects)
# text(qqnorm(effects)$x,qqnorm(effects)$y, names(effects))
# qqline(effects)
```