ISYE 6501 HW9

Chengqi

2019/10/16

Question 12.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a design of experiments approach would be appropriate.

Suppose we are the record company running the hip-hop show Rhythm and Flow, we may want to know which candidate of the champion can bring greater benefits to the company, so that the record company will be able to cheat and manipulate the results of the competition behind the scenes.

Since we may have many potential championship candidates, and other candidates may also bring profit to the company, we don't want to lose value. In this case, we can use the Multi Armed Bandit approach. During the last episode before the final, the company can randomly ask an audience to buy virtual currency in order to show his or her support to one of the candidates. Then we can renew the score of this particular candidate, and ask another audience rating another candidate. First, every candidate has equal probability of selecting. But as we keep renewing each candidate's score, the probability of appearance of an individual candidate changes. Finally we are able to have the result of who is likely to be the best. Moreover, if every candidate have an equal probability of appearing, people are less likely to buy virtual currency to support a less popular candidate. Thus we successfully prevent losing money.

Question 12.2

To determine the value of 10 different yes/no features to the market value of a house (large yard, solar roof, etc.), a real estate agent plans to survey 50 potential buyers, showing a fictitious house with different combinations of features. To reduce the survey size, the agent wants to show just 16 fictitious houses. Use R's FrF2 function (in the FrF2 package) to find a fractional factorial design for this experiment: what set of features should each of the 16 fictitious houses have? Note: the output of FrF2 is "1" (include) or "-1" (don't include) for each feature.

We use the FrF2 function to choose 16 different combinations among 10 factors

```r
#First, we should load the FrF2 Package
library(FrF2)
```

```
## Loading required package: DoE.base
```

```
## Loading required package: grid
```

```
## Loading required package: conf.design
```

```
## Registered S3 method overwritten by 'DoE.base':
##   method           from
##   factorize.factor conf.design
```

```
##
## Attaching package: 'DoE.base'
```

```
## The following objects are masked from 'package:stats':
##
##     aov, lm
```

```
## The following object is masked from 'package:graphics':
##
##     plot.design
```

```
## The following object is masked from 'package:base':
##
##     lengths
```

```r
set.seed(123)
#We choose 10 different factors that affects the value of a house
factors = c('large yard', 'solar roof','dog park','free coffee','with f
urniture'
            ,'swimming pool','grocery nearby','school nearby','garage',
'gym')
#We have 10 different features and we want to test 16 different combina
tions
conbination <- FrF2(nruns = 16,nfactors = 10,factor.names = factors)
#Finally we have the results: in the following chart, 1 means we choose
 that factor, and -1 refers to not included of a certain factor. We can
 use the following 16 combinations:
conbination
```

```
##    large.yard solar.roof dog.park free.coffee with.furniture swimmin
g.pool
## 1          -1          1        1           1             -1
    -1
## 2           1          1        1           1              1
     1
## 3          -1          1       -1          -1             -1
     1
## 4           1         -1        1           1             -1
     1
```

```
## 5           1         -1       -1           1              -1
    -1
## 6           1         -1       -1          -1              -1
    -1
## 7           1         -1        1          -1              -1
     1
## 8          -1         -1        1          -1               1
    -1
## 9           1          1       -1          -1               1
    -1
## 10          1          1       -1           1               1
    -1
## 11         -1          1        1          -1              -1
    -1
## 12         -1         -1       -1          -1               1
     1
## 13         -1          1       -1           1              -1
     1
## 14         -1         -1        1           1               1
    -1
## 15          1          1        1          -1               1
     1
## 16         -1         -1       -1           1               1
     1
##    grocery.nearby school.nearby garage gym
## 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
## 9              -1            -1      1   1
## 10             -1             1     -1  -1
## 11              1             1     -1   1
## 12              1             1     -1   1
## 13             -1            -1     -1   1
## 14             -1            -1     -1   1
## 15              1            -1     -1  -1
## 16              1            -1      1  -1
## class=design, type= FrF2
```

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values. 1. Use the mean/mode imputation method to impute values for the missing data. 2. Use regression to impute values for the missing data. 3. Use regression with perturbation to impute values for the missing data. 4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using (1) the data sets from questions 1,2,3; (2) the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values. 1. Use the mean/mode imputation method to impute values for the missing data.

```
#load data
data <- read.table('C:\\Users\\huangchengqi\\Desktop\\MS SCE\\19Fall\\ISYE6501\\hw9\\breast-cancer-wisconsin.data.txt', header = FALSE, sep = ",")
```

First of all, we need to find the missing data:

```
location <- c()
jset <- c()
iset <- c()
for (i in 1:699) {
  for (j in 1:11){
    if (data[i,j] == '?'){
      location <- c(location,c(i,j))
      iset <- c(iset, i)
      jset <- c(jset, j)
    }
  }
}
iset
```

```
##  [1]  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
```

```
jset
```

```
##  [1] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
```

```
location
```

```
##  [1]  24   7  41   7 140   7 146   7 159   7 165   7 236   7 250   7
 276
## [18]   7 293   7 295   7 298   7 316   7 322   7 412   7 618   7
```

```
#now we have the location of missing values.It is shown that only the 7th column has missing value.
data[which(data$V7 == "?"),]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  ?  7  3   1   4
## 41  1096800  6  6  6  9  6  ?  7  8   1   2
## 140 1183246  1  1  1  1  1  ?  2  1   1   2
```

```
## 146 1184840  1  1  3  1  2  ?  2  1   1   2
## 159 1193683  1  1  2  1  3  ?  1  1   1   2
## 165 1197510  5  1  1  1  2  ?  3  1   1   2
## 236 1241232  3  1  4  1  2  ?  3  1   1   2
## 250  169356  3  1  1  1  2  ?  3  1   1   2
## 276  432809  3  1  3  1  2  ?  2  1   1   2
## 293  563649  8  8  8  1  2  ?  6 10   1   4
## 295  606140  1  1  1  1  2  ?  2  1   1   2
## 298   61634  5  4  3  1  2  ?  2  3   1   2
## 316  704168  4  6  5  6  7  ?  4  9   1   2
## 322  733639  3  1  1  1  2  ?  3  1   1   2
## 412 1238464  1  1  1  1  1  ?  2  1   1   2
## 618 1057067  1  1  1  1  1  ?  1  1   1   2

missing <- which(data$V7 == "?", arr.ind = TRUE)
```

Now we can use the imputation method to deal with missing data. First is the mean/mode impotation method. These approaches requires us to use the mean/mode of the existing values in the 7th column as the missing values.

```
#mean/mode imputation method
#mean
data2 <- data
data2[data2 == '?'] <- NA
data3 <- na.omit(data2)
mean <- mean(as.numeric(data3[,7]))
data_mean <- data
data_mean[data_mean == '?'] <- as.integer(mean)
data_mean[missing,]

##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  3  7  3   1   4
## 41  1096800  6  6  6  9  6  3  7  8   1   2
## 140 1183246  1  1  1  1  1  3  2  1   1   2
## 146 1184840  1  1  3  1  2  3  2  1   1   2
## 159 1193683  1  1  2  1  3  3  1  1   1   2
## 165 1197510  5  1  1  1  2  3  3  1   1   2
## 236 1241232  3  1  4  1  2  3  3  1   1   2
## 250  169356  3  1  1  1  2  3  3  1   1   2
## 276  432809  3  1  3  1  2  3  2  1   1   2
## 293  563649  8  8  8  1  2  3  6 10   1   4
## 295  606140  1  1  1  1  2  3  2  1   1   2
## 298   61634  5  4  3  1  2  3  2  3   1   2
## 316  704168  4  6  5  6  7  3  4  9   1   2
## 322  733639  3  1  1  1  2  3  3  1   1   2
## 412 1238464  1  1  1  1  1  3  2  1   1   2
## 618 1057067  1  1  1  1  1  3  1  1   1   2

#mode
#we first write our own function to find the mode:
mode_function <- function(x)
```

```
{
  return(as.numeric(names(table(x))[table(x) == max(table(x))]))
}
mode <- as.numeric(mode_function(data3[,7]))
data_mode <- data
data_mode[data_mode == '?'] <- mode
data_mode[missing,]
```

```
##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  1  7  3   1   4
## 41  1096800  6  6  6  9  6  1  7  8   1   2
## 140 1183246  1  1  1  1  1  1  2  1   1   2
## 146 1184840  1  1  3  1  2  1  2  1   1   2
## 159 1193683  1  1  2  1  3  1  1  1   1   2
## 165 1197510  5  1  1  1  2  1  3  1   1   2
## 236 1241232  3  1  4  1  2  1  3  1   1   2
## 250  169356  3  1  1  1  2  1  3  1   1   2
## 276  432809  3  1  3  1  2  1  2  1   1   2
## 293  563649  8  8  8  1  2  1  6 10   1   4
## 295  606140  1  1  1  1  2  1  2  1   1   2
## 298   61634  5  4  3  1  2  1  2  3   1   2
## 316  704168  4  6  5  6  7  1  4  9   1   2
## 322  733639  3  1  1  1  2  1  3  1   1   2
## 412 1238464  1  1  1  1  1  1  2  1   1   2
## 618 1057067  1  1  1  1  1  1  1  1   1   2
```

2. Use regression to impute values for the missing data. First of all, we use all other attributes as predictors. Since the first attribute is id number, we do not want to include.moreover, we do not want to include the missing data to run a regression model. So first we should modify the original data:

```
data_modified <- data[-missing,2:11]
data_modified$V7 <- as.integer(data_modified$V7)
model_reg <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10+V11, data = data_modified)
summary(model_reg)
```

```
##
## Call:
## lm.default(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 +
##     V10 + V11, data = data_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8188 -0.5655 -0.4231 -0.2615  7.4773
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.29992    0.28479   1.053 0.292665
## V2          -0.02285    0.03642  -0.627 0.530673
## V3           0.04746    0.06188   0.767 0.443349
## V4           0.04296    0.06022   0.713 0.475812
```

```
## V5          -0.12800    0.03792  -3.376 0.000779 ***
## V6           0.01263    0.05077   0.249 0.803674
## V8          -0.02823    0.04899  -0.576 0.564644
## V9           0.07873    0.03649   2.158 0.031308 *
## V10          0.00666    0.04797   0.139 0.889610
## V11          1.07883    0.16373   6.589 8.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.839 on 673 degrees of freedom
## Multiple R-squared:  0.2791, Adjusted R-squared:  0.2695
## F-statistic: 28.95 on 9 and 673 DF,  p-value: < 2.2e-16
```

We may don't want to include all the attributes as predictors. Thus we use the stepwise regression to find the best combination of predictors:

```
library(MASS)
stepAIC(model_reg, direction='both', steps=500)

## Start:  AIC=841.83
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V11
##
##        Df Sum of Sq    RSS    AIC
## - V10   1     0.065 2275.1 839.85
## - V6    1     0.209 2275.3 839.89
## - V8    1     1.123 2276.2 840.17
## - V2    1     1.330 2276.4 840.23
## - V4    1     1.721 2276.8 840.35
## - V3    1     1.989 2277.0 840.43
## <none>              2275.1 841.83
## - V9    1    15.738 2290.8 844.54
## - V5    1    38.520 2313.6 851.30
## - V11   1   146.761 2421.8 882.53
##
## Step:  AIC=839.85
## V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V11
##
##        Df Sum of Sq    RSS    AIC
## - V6    1     0.268 2275.4 837.93
## - V8    1     1.183 2276.3 838.20
## - V2    1     1.300 2276.4 838.24
## - V4    1     1.719 2276.8 838.36
## - V3    1     2.029 2277.2 838.46
## <none>              2275.1 839.85
## + V10   1     0.065 2275.1 841.83
## - V9    1    16.232 2291.3 842.70
## - V5    1    38.721 2313.8 849.38
## - V11   1   146.699 2421.8 880.53
##
## Step:  AIC=837.93
```

```
## V7 ~ V2 + V3 + V4 + V5 + V8 + V9 + V11
##
##        Df Sum of Sq    RSS    AIC
## - V8    1     1.173 2276.6 836.28
## - V2    1     1.307 2276.7 836.32
## - V4    1     1.815 2277.2 836.47
## - V3    1     2.568 2277.9 836.70
## <none>              2275.4 837.93
## + V6    1     0.268 2275.1 839.85
## + V10   1     0.124 2275.3 839.89
## - V9    1    16.974 2292.4 841.01
## - V5    1    38.476 2313.9 847.38
## - V11   1   149.035 2424.4 879.26
##
## Step:  AIC=836.28
## V7 ~ V2 + V3 + V4 + V5 + V9 + V11
##
##        Df Sum of Sq    RSS    AIC
## - V2    1     1.270 2277.8 834.66
## - V4    1     1.693 2278.2 834.79
## - V3    1     2.095 2278.7 834.91
## <none>              2276.6 836.28
## + V8    1     1.173 2275.4 837.93
## + V6    1     0.258 2276.3 838.20
## + V10   1     0.200 2276.4 838.22
## - V9    1    16.036 2292.6 839.08
## - V5    1    42.331 2318.9 846.86
## - V11   1   150.988 2427.6 878.14
##
## Step:  AIC=834.66
## V7 ~ V3 + V4 + V5 + V9 + V11
##
##        Df Sum of Sq    RSS    AIC
## - V4    1     1.426 2279.2 833.09
## - V3    1     1.946 2279.8 833.25
## <none>              2277.8 834.66
## + V2    1     1.270 2276.6 836.28
## + V8    1     1.137 2276.7 836.32
## + V6    1     0.264 2277.6 836.58
## + V10   1     0.145 2277.7 836.62
## - V9    1    16.113 2293.9 837.48
## - V5    1    41.371 2319.2 844.96
## - V11   1   165.178 2443.0 880.48
##
## Step:  AIC=833.09
## V7 ~ V3 + V5 + V9 + V11
##
##        Df Sum of Sq    RSS    AIC
## <none>              2279.2 833.09
## - V3    1     8.535 2287.8 833.64
```

```
## + V4    1     1.426 2277.8 834.66
## + V8    1     1.029 2278.2 834.78
## + V2    1     1.003 2278.2 834.79
## + V6    1     0.348 2278.9 834.98
## + V10   1     0.156 2279.1 835.04
## - V9    1    17.665 2296.9 836.36
## - V5    1    40.801 2320.1 843.21
## - V11   1   184.585 2463.8 884.28

##
## Call:
## lm.default(formula = V7 ~ V3 + V5 + V9 + V11, data = data_modified)
##
## Coefficients:
## (Intercept)           V3           V5           V9          V11
##     0.29956      0.07021     -0.12774      0.08062      1.04675
```

```
model_reg_2 <- lm(V7 ~ V3 + V5 + V9 + V11, data = data_modified)
summary(model_reg_2)
```

```
##
## Call:
## lm.default(formula = V7 ~ V3 + V5 + V9 + V11, data = data_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7059 -0.5566 -0.4161 -0.2884  7.4525
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.29956    0.27503   1.089 0.276450
## V3           0.07021    0.04407   1.593 0.111547
## V5          -0.12774    0.03667  -3.484 0.000526 ***
## V9           0.08062    0.03517   2.292 0.022191 *
## V11          1.04675    0.14126   7.410 3.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.834 on 678 degrees of freedom
## Multiple R-squared:  0.2778, Adjusted R-squared:  0.2735
## F-statistic:  65.2 on 4 and 678 DF,  p-value: < 2.2e-16
```

We can also use the mice package to get the missing values. The results are the same.

```
library(mice)
```

```
## Loading required package: lattice

##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:DoE.base':
##
##     make.formulas

## The following objects are masked from 'package:base':
##
##     cbind, rbind

imp1 <- mice(data_modified, m = 5)

##
##  iter imp variable
##   1   1
##   1   2
##   1   3
##   1   4
##   1   5
##   2   1
##   2   2
##   2   3
##   2   4
##   2   5
##   3   1
##   3   2
##   3   3
##   3   4
##   3   5
##   4   1
##   4   2
##   4   3
##   4   4
##   4   5
##   5   1
##   5   2
##   5   3
##   5   4
##   5   5

fitm <- with(imp1, lm(V7 ~ V3 + V5 + V9 + V11))
summary(fitm)

## # A tibble: 25 x 5
##    term        estimate std.error statistic  p.value
##    <chr>          <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)   0.300     0.275       1.09 2.76e- 1
##  2 V3            0.0702    0.0441      1.59 1.12e- 1
##  3 V5           -0.128     0.0367     -3.48 5.26e- 4
##  4 V9            0.0806    0.0352      2.29 2.22e- 2
##  5 V11           1.05      0.141       7.41 3.77e-13
##  6 (Intercept)   0.300     0.275       1.09 2.76e- 1
##  7 V3            0.0702    0.0441      1.59 1.12e- 1
```

```
##  8 V5           -0.128     0.0367     -3.48 5.26e- 4
##  9 V9            0.0806    0.0352      2.29 2.22e- 2
## 10 V11           1.05      0.141       7.41 3.77e-13
## # ... with 15 more rows

pool(fitm)

## Class: mipo    m = 5
##                 estimate        ubar b           t dfcom       df riv
## (Intercept)  0.29955991 0.075639261 0 0.075639261   678 675.9401   0
## V3           0.07021277 0.001941818 0 0.001941818   678 675.9401   0
## V5          -0.12773596 0.001344379 0 0.001344379   678 675.9401   0
## V9           0.08062070 0.001236894 0 0.001236894   678 675.9401   0
## V11          1.04674607 0.019954853 0 0.019954853   678 675.9401   0
##             lambda         fmi
## (Intercept)      0 0.002945768
## V3               0 0.002945768
## V5               0 0.002945768
## V9               0 0.002945768
## V11              0 0.002945768

summary(pool(fitm))

##                 estimate  std.error statistic       df      p.value
## (Intercept)  0.29955991 0.27502593  1.089206 675.9401 2.764512e-01
## V3           0.07021277 0.04406606  1.593352 675.9401 1.115486e-01
## V5          -0.12773596 0.03666578 -3.483793 675.9401 5.262546e-04
## V9           0.08062070 0.03516950  2.292347 675.9401 2.219202e-02
## V11          1.04674607 0.14126165  7.409981 675.9401 3.783640e-13
```

As we already have a regression model, we are able to get the missing values:

```
missing_value <- predict(model_reg_2, data2[missing,])
View(missing_value)
#missing value should all be integers:
missing_value_reg <- round(missing_value)
View(missing_value_reg)
#finally we have the whole data set using regression imputation method:
data_reg <- data
data_reg[missing,7] <- missing_value_reg
data_reg[missing,]

##          V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2  5  7  3   1   4
## 41  1096800  6  6  6  9  6  2  7  8   1   2
## 140 1183246  1  1  1  1  1  2  2  1   1   2
## 146 1184840  1  1  3  1  2  2  2  1   1   2
## 159 1193683  1  1  2  1  3  2  1  1   1   2
## 165 1197510  5  1  1  1  2  2  3  1   1   2
## 236 1241232  3  1  4  1  2  2  3  1   1   2
## 250  169356  3  1  1  1  2  2  3  1   1   2
```

```
## 276  432809  3  1  3  1  2  2  2  1   1   2
## 293  563649  8  8  8  1  2  6  6 10   1   4
## 295  606140  1  1  1  1  2  2  2  1   1   2
## 298   61634  5  4  3  1  2  3  2  3   1   2
## 316  704168  4  6  5  6  7  3  4  9   1   2
## 322  733639  3  1  1  1  2  2  3  1   1   2
## 412 1238464  1  1  1  1  1  2  2  1   1   2
## 618 1057067  1  1  1  1  1  2  1  1   1   2
```

3. Use regression with perturbation to impute values for the missing data.

We can perturb the predictions for missing V7 values with a random normal distriubtion. The mean and standard deviation of the distribution is decided by existing predicted values we calculated in the regression imputation method.

```
#create random values which fits normal distribution
missing_value_reg_2 <- rnorm(16, missing_value, sd(missing_value))
View(missing_value_reg_2)
#missing value should all be integers:
missing_value_reg_pret <- round(missing_value_reg_2)
data_reg_prep <- data
#finally we have the whole data set using regression perturbation imput
ation method:
data_reg_prep[missing,7] <- missing_value_reg_pret
```

```
## Warning in `[<-.factor`(`*tmp*`, iseq, value = c(4, 4, 3, 3, 3, 2, 4,
 3, :
## invalid factor level, NA generated
```

```
data_reg_prep[missing,]
```

```
##          V1 V2 V3 V4 V5 V6   V7 V8 V9 V10 V11
## 24  1057013  8  4  5  1  2    4  7  3   1   4
## 41  1096800  6  6  6  9  6    4  7  8   1   2
## 140 1183246  1  1  1  1  1    3  2  1   1   2
## 146 1184840  1  1  3  1  2    3  2  1   1   2
## 159 1193683  1  1  2  1  3    3  1  1   1   2
## 165 1197510  5  1  1  1  2    2  3  1   1   2
## 236 1241232  3  1  4  1  2    4  3  1   1   2
## 250  169356  3  1  1  1  2    3  3  1   1   2
## 276  432809  3  1  3  1  2    1  2  1   1   2
## 293  563649  8  8  8  1  2    6  6 10   1   4
## 295  606140  1  1  1  1  2    2  2  1   1   2
## 298   61634  5  4  3  1  2    2  2  3   1   2
## 316  704168  4  6  5  6  7    3  4  9   1   2
## 322  733639  3  1  1  1  2    1  3  1   1   2
## 412 1238464  1  1  1  1  1    2  2  1   1   2
## 618 1057067  1  1  1  1  1    2  1  1   1   2
```