ISyE 6501 HW9

October 17, 2019

Question 12.1

The insomnia epidemic marring the United States and many other developed nations qualifies as one of the defining problems of our time. Practicioners' guidelines list, among the many prescriptions, those of limiting caffeinated beverages, tapering off blue-light emitting devices after dusk, setting bedroom temperature quite low (in the range 60 to 67°F), having regular exercise. Unfortunately, the bulk of the academic endeavor takes on much more hardcore medical questions, such as the impact of the many available drugs on sleep. The reliability of the more behavioral guidelines is often left to the speculations of non-academic sources, which provide little help to those who want to fix their sleep problems in an evidence-based fashion.
To tackle this problem, we suggest to implement a DOE approach using the aforementioned inputs as controllable factors: *a)* use of caffeinated beverages; *b)* exposure to blue-light emitting devices; *c)* bedroom temperature; *d)* physical exercise during the day. These can be easily operationalized; for instance, input *c)* can be set at 60°F, 67°F, and 75°F. The responses for the experiment will be Sleep Onset Latency (SOL) and time asleep, both measured in minutes. The uncontrollable factors are many, ranging from anxiety levels of the tested subjects, lifestyle, diet, and job type. To circumscribe those, we will dig existing research and might also host a focus group of selected poor sleepers.
Last, the experiment setup. We will not resort to complete randomization of our study subjects and will instead block subjects into groups that represent the most important uncontrollable factors in the population. For instance, we suspect that students in their first year of college share common uncontrollable factors that also affect our response variables such as anxiety levels, lifestyle, and diet. Therefore, we will want to equally distribute this group across our experimental setups. By the same token, mid-career professionals in high-paying jobs, and senior workers in low-skilled occupations might be targeted. This characterization of the experimental population is far from exhaustive and only serves demonstration purposes. It shows how blocking can be leveraged by analysts to deal with uncontrollable factors.
Lastly, interactions might be considered between inputs *a)* to *d)*. For instance, it is believable that exposure to blue-light emitting devices and bedroom temperature contribute to overall sleep hygiene, and interact with each other. When exposure to blue lights is high, higher bedroom temperatures might be even more detrimental to sleep onset and maintenance. This is the equivalent to interaction terms in regression, and is, of course, one of the experimental hypotheses that we will carefully write down based on existing research and our hunches as analysts.

Question 12.2

The fractional factorial design is shown below. The output of FrF2 has been recoded to take on the values "Yes" and "No". Each row represents one of the fictitious houses that the agent will show to her customers, and its characteristics.

```
library(FrF2)
```

```
v1 <- rep("Factor", 10)
v2 <- c(1:10)
factors = paste0(v1, v2)
FrF2(16, 10, default.levels = c("No", "Yes"), factor.names = factors)
```

```
##    Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8 Factor9
```

```
## 1       No     Yes      No      No      No     Yes      No     Yes     Yes
## 2      Yes      No      No     Yes      No      No     Yes     Yes     Yes
## 3       No      No     Yes     Yes     Yes      No      No      No      No
## 4       No      No     Yes      No     Yes      No      No     Yes     Yes
## 5      Yes     Yes      No     Yes     Yes      No      No     Yes      No
## 6      Yes     Yes     Yes      No     Yes     Yes     Yes      No      No
## 7       No     Yes     Yes     Yes      No      No     Yes      No     Yes
## 8       No     Yes     Yes      No      No      No     Yes     Yes      No
## 9      Yes      No      No      No      No      No     Yes      No      No
## 10      No      No      No      No     Yes     Yes     Yes     Yes      No
## 11     Yes     Yes      No      No     Yes      No      No      No     Yes
## 12     Yes     Yes     Yes     Yes     Yes     Yes     Yes     Yes     Yes
## 13      No     Yes      No     Yes      No     Yes      No      No      No
## 14     Yes      No     Yes     Yes      No     Yes      No     Yes      No
## 15      No      No      No     Yes     Yes     Yes     Yes      No     Yes
## 16     Yes      No     Yes      No      No     Yes      No      No     Yes
##    Factor10
## 1        No
## 2       Yes
## 3       Yes
## 4        No
## 5        No
## 6        No
## 7        No
## 8       Yes
## 9        No
## 10      Yes
## 11      Yes
## 12      Yes
## 13      Yes
## 14       No
## 15       No
## 16      Yes
## class=design, type= FrF2
```

Question 14.1

```r
# loading data
data = read.table("breast-cancer-wisconsin.data.txt", header = FALSE,
    sep = ",")

# loading packages 'caret', 'mice', 'stepAIC'
library(caret)
library(mice)
library(MASS)

# loading packages for mode calculation
library("BiocManager")
library(modeest)

# avoiding scientific notation
options(scipen = 4, digits = 3)
```

Data Preparation

Before looking at missing cases, we performed some data handling procedures to make our data more tractable. First, we gave appropriate column names corresponding to the variables in the dataset. Second, we recoded the dependent variable *tumor* to a factor variable taking on 0 for negatives and 1 for positives, and the variable *nuclei* to a continuous variable. The original coding as factor does not make statistical sense because the variable captures the size of the nuclei, hence an increase/decrease in its value represents an increase/decrease in size. Also, we created four destination columns for the imputated data based off of the variable with missing cases *nuclei*, and an indicator variable for missing cases coded as 1 for data points that have missing values and 0 for complete cases.

```
# renaming columns
colnames(data) <- c("ID", "clump_thick", "size_unif", "shape_unif", "adhesion",
    "epith_size", "nuclei", "chromatin", "nucleoli", "mitoses", "tumor")

# recoding response
data$tumor[data$tumor == 4] <- 1
data$tumor[data$tumor == 2] <- 0
data$tumor = as.factor(data$tumor)

# recoding factor 'nuclei' as continuous
data$nuclei <- as.numeric(as.character(data$nuclei))

# creating indicator for missing cases
data$missing <- as.factor(ifelse(is.na(data$nuclei), 1, 0))

# creating destination columns
data$nuclei_mean = data$nuclei
data$nuclei_mode = data$nuclei
data$nuclei_MICE.nob = data$nuclei
data$nuclei_MICE.boot = data$nuclei
```

The data has missing cases, and these represent about 2.3% of total observations. The md.pattern() command from the 'mice' package is very handy and displays patterns of missingness. The output table informs us that there are 683 observations with full cases and 16 observations for which the value of *nuclei* is missing.

```
# looking for missing cases
any(is.na(data))
```

```
## [1] TRUE
```

```
# percentage of missing cases
exclvar <- names(data) %in% c("ID", "nuclei_mean", "nuclei_mode", "nuclei_MICE.nob",
    "nuclei_MICE.boot", "missing")
nrow(data[!complete.cases(data[!exclvar]), ])/nrow(data) * 100
```

```
## [1] 2.29
```

```
# pattern of missing cases
md.pattern(data[!exclvar], plot = FALSE)
```

```
##     clump_thick size_unif shape_unif adhesion epith_size chromatin nucleoli
## 683           1         1          1        1          1         1        1
## 16            1         1          1        1          1         1        1
##               0         0          0        0          0         0        0
##     mitoses tumor nuclei
## 683       1     1      1  0
## 16        1     1      0  1
```

```
##           0     0     16 16
```

Mean/mode Imputation

Mean/mode imputation can be easily performed manually without the aid of custom-made R packages. All we need to do is to calculate the mean/mode for the variable *nuclei* and impute missing cases with these values. The mean value is 3.54 and the mode is 1; these are imputed to the missing cases in the respective destination columns created before.

```
# mean imputation
mean(data$nuclei_mean, na.rm = TRUE)
```

```
## [1] 3.54
```

```
data$nuclei_mean[is.na(data$nuclei_mean)] <- mean(data$nuclei_mean, na.rm = TRUE)

# mode imputation
mfv(data$nuclei_mode, na.rm = TRUE)
```

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

```
data$nuclei_mode[is.na(data$nuclei_mode)] <- mfv(data$nuclei_mode, na.rm = TRUE)
```

MICE Imputation

Multiple Imputation by Chained Equations (MICE) is nothing but an application of regression. The variable with missing cases is regressed against a matrix of regressors chosen by the analyst, and predicted values are imputed to the missing entries. There are no one-size-fits-all rules on how to specify imputation equations other than the recommended practice of including the independent variable of the final model; in our case, the variable *tumor*. Sometimes, few regressors are necessary to obtain reasonable estimates of the missing values. For instance, when imputing missing cases on a variable coding a student's parental education is often times necessary to specify parental income alone. Other times, we will instead recruit all the information stored in the dataset, in which case we would specify an imputation equation that includes all the full-case variables at our disposal. This is the approach we followed in this homework.
The problem with deterministic MICE is that it does not account for uncertainty in the data. We can understand this better by way of an example. Let's suppose that we have student-level data with four variables, one for parental education, one for parental income, one for parental support (coded ordinally from "very low" to "very high"), and the achievement score on the GRE quant score. Let's imagine that we have missing data on the parental support variable, and that we use MICE to impute. What would happen is that two students that have the same levels of parental education and parental income and the same GRE quant score would be mechanically predicted to have the same level of parental support no matter what. This example provides a rationale for the use of perturbation methods in MICE. Without going into the technicalities, perturbation methods introduce some randomness in the imputations. The two indistinguishable students of ours would now be imputed two different levels of parental support, even if their "predicted" support is the same.

```
# MICE imputation (deterministic)
predMatrix <- make.predictorMatrix(data[-c((ncol(data) - 4):ncol(data))])
mice.nob <- mice(data = data[-c((ncol(data) - 4):ncol(data))], m = 1,
    predMatrix, where = is.na(data[-c((ncol(data) - 4):ncol(data))]),
    method = "norm.nob", post = NULL, maxit = 50, printFlag = FALSE, seed = 123)

# displaying imputations
mice.nob$imp$nuclei
```

```
##          1
## 24   8.829
```

```
## 41   4.196
## 140  0.297
## 146  2.042
## 159  2.556
## 165  2.142
## 236  0.628
## 250  3.273
## 276  4.047
## 293  6.986
## 295  1.592
## 298  1.035
## 316  6.407
## 322  2.022
## 412  0.863
## 618 -3.958
```

```
# obtaining imputed data
imputed <- complete(mice.nob, action = "broad", include = F)

# filling out detsination column with imputed values
data[, "nuclei_MICE.nob"] <- imputed$nuclei.1
rm(imputed)
```

```
# MICE imputation (with perturbation)
mice.boot <- mice(data = data[-c((ncol(data) - 3):ncol(data))], m = 1,
    predMatrix, where = is.na(data[-c((ncol(data) - 3):ncol(data))]),
    method = "norm.boot", post = NULL, maxit = 50, printFlag = FALSE,
    seed = 123)

# displaying imputations
mice.boot$imp$nuclei
```

```
##           1
## 24   6.8619
## 41   4.2839
## 140  0.1974
## 146 -0.7658
## 159 -2.5270
## 165 -0.0743
## 236  1.8758
## 250  2.3467
## 276  1.3955
## 293  6.3796
## 295  2.5181
## 298 -2.5947
## 316  0.7386
## 322 -1.3225
## 412  2.6477
## 618 -3.6210
```

```
# obtaining imputed data
imputed <- complete(mice.boot, action = "broad", include = F)

# filling out detsination column with imputed values
data[, "nuclei_MICE.boot"] <- imputed$nuclei.1
```

```r
rm(imputed)
```

Assessing Imputation Performance

To assess imputation performance, we trained a logistic regression model on six different datasets using the different methods for imputation: mean, mode, MICE (deterministic), MICE (with perturbation), listwise deletion of missing cases, and dummy variable.
A couple (or triple) of remarks before getting started. Since we have the few of 16 missing cases, we did not expect the predictive performance of the model to change meaningfully across the different datasets. Also, the complete cases variables are very predictive of the outcome (i.e. *tumor*) which further dampens the value of performing an imputation. When we first trained a stepwise logistic regression model, we failed to retrieve any difference in performance. Predictive accuracy was the exact same for every model, and extremely high (i.e. 96.1%). To shuffle things around a bit, we trained a reduced logistic model with only three regressors other than the variable with incomplete cases *nuclei*.
The next asides cover the listwise deletion and the dummy variable approaches. Listwise deletion is by far the less computationally-intensive and most straightforward of the ways to handle missing cases. However, it is also the less recommended for virtually any occurrence exception made for data that are Missing Completely at Random (MCAR). In fact, there are usually patterns among missing cases that affect the outcome of interest and therefore bias the results. Revamping our student-level data example, it is believable that poorer students are less likely to report parental income while at the same time being more likely to underperform on the GRE. Listwise deletion of missing cases would remove significant information from the data and substantially bias the results.
The dummy variable approach (Cohen & Cohen, 1975) consists of a two-step procedure: *i)* an indicator variable is created that takes on 1s for missing cases, 0s for complete cases; *ii)* arbitrary values are imputed to missing cases, most often 0 or the mean of the observed values (we used 0). Whether the associated coefficient on the indicator variable is statistically significant informs on the potential bias caused by the omitted information. While not intuitively appealing, this approach is much more statistically sound than listwise deletion.
The last aside goes to the train/test split. All the missing/imputed cases were included in the training datasets, and only complete cases where chosen to populate the test data. In fact, it is the purpose of the homework assignment to see how different approaches for handling missing data lead to different estimation results, and we really want to have all of our missing cases in the train data.

```r
# excluding imputed cases from partition
missing <- data[which(is.na(data$nuclei)), ]
full <- data[-which(is.na(data$nuclei)), ]

# sampling full dataset
sample <- sample.int(n = nrow(full), size = floor(0.7 * nrow(full)), replace = FALSE)
train <- full[sample, ]
test <- full[-sample, ]

# including imputed cases in train dataset
train <- rbind(missing, train)

# generating training sets for mean, mode, and MICE imputation
trainMean <- subset(train, select = -c(nuclei, nuclei_mode, nuclei_MICE.nob,
    nuclei_MICE.boot, missing))
trainMode <- subset(train, select = -c(nuclei, nuclei_mean, nuclei_MICE.nob,
    nuclei_MICE.boot, missing))
trainMICE.nob <- subset(train, select = -c(nuclei, nuclei_mean, nuclei_mode,
    nuclei_MICE.boot, missing))
trainMICE.boot <- subset(train, select = -c(nuclei, nuclei_mean, nuclei_mode,
    nuclei_MICE.nob, missing))
```

```
# generating training set with listwise deletion of missing cases
trainDrop <- subset(train, select = -c(nuclei_mean, nuclei_mode, nuclei_MICE.nob,
    nuclei_MICE.boot, missing))
trainDrop <- na.omit(trainDrop)

# generating training set with indicator for missing cases
trainDummy <- subset(train, select = -c(nuclei_mean, nuclei_mode, nuclei_MICE.nob,
    nuclei_MICE.boot))
trainDummy$nuclei[is.na(trainDummy$nuclei)] <- 0
```

As mentioned before, the logistic model is used for demonstration purposes, and has been limited to three explanatory variables other than the variable with missing cases *nuclei*. The fit is extremely high with three models (i.e. mean, mode, and dummy variable) peaking at 96.6% predictive accuracy and the three remaining models (i.e. MICE deterministic, MICE with perturbation, listwise deletion) entering at 96.1%. The difference is not substantial, and we must say that all models fare pretty good. However, we should bear in mind that only one of the variables had missing cases, and these were the few of 16 observations. We are not that familiar with medical research; however, we can confidently say that economic and business analytics are rarely blessed with such glory. Most of the times, analysts deal with multiple variables with missing values, and a much lower proportion of complete cases.

```
# testing performance for mean imputation
model <- glm(tumor ~ clump_thick + adhesion + mitoses + nuclei_mean, data = trainMean,
    family = binomial(link = "logit"))

# accuracy and confusion matrix
predMean <- predict(model, test, type = "response")
predMean <- as.factor(ifelse(predMean > 0.5, 1, 0))
mean(predMean == test$tumor)
```

```
## [1] 0.966
```

```
table(predMean, test$tumor)
```

```
##
## predMean   0   1
##        0 137   6
##        1   1  61
```

```
# testing performance for mode imputation
model <- glm(tumor ~ clump_thick + adhesion + mitoses + nuclei_mode, data = trainMode,
    family = binomial(link = "logit"))

# accuracy and confusion matrix
predMode <- predict(model, test, type = "response")
predMode <- as.factor(ifelse(predMode > 0.5, 1, 0))
mean(predMode == test$tumor)
```

```
## [1] 0.966
```

```
table(predMode, test$tumor)
```

```
##
## predMode   0   1
##        0 136   5
##        1   2  62
```

```r
# testing performance for MICE imputation (deterministic)
model <- glm(tumor ~ clump_thick + adhesion + mitoses + nuclei_MICE.nob,
    data = trainMICE.nob, family = binomial(link = "logit"))

# accuracy and confusion matrix
predMICE.nob <- predict(model, test, type = "response")
predMICE.nob <- as.factor(ifelse(predMICE.nob > 0.5, 1, 0))
mean(predMICE.nob == test$tumor)
```

```
## [1] 0.961
```

```r
table(predMICE.nob, test$tumor)
```

```
##
## predMICE.nob   0   1
##            0 137   7
##            1   1  60
```

```r
# testing performance for MICE imputation (with perturbation)
model <- glm(tumor ~ clump_thick + adhesion + mitoses + nuclei_MICE.boot,
    data = trainMICE.boot, family = binomial(link = "logit"))

# accuracy and confusion matrix
predMICE.boot <- predict(model, test, type = "response")
predMICE.boot <- as.factor(ifelse(predMICE.boot > 0.5, 1, 0))
mean(predMICE.boot == test$tumor)
```

```
## [1] 0.961
```

```r
table(predMICE.boot, test$tumor)
```

```
##
## predMICE.boot   0   1
##             0 137   7
##             1   1  60
```

```r
# testing performance for listwise deletion method
model <- glm(tumor ~ clump_thick + adhesion + mitoses + nuclei, data = trainDrop,
    family = binomial(link = "logit"))

# accuracy and confusion matrix
predDrop <- predict(model, test, type = "response")
predDrop <- as.factor(ifelse(predDrop > 0.5, 1, 0))
mean(predDrop == test$tumor)
```

```
## [1] 0.961
```

```r
table(predDrop, test$tumor)
```

```
##
## predDrop   0   1
##        0 137   7
##        1   1  60
```

```r
# testing performance for dummy variable deletion
model <- glm(tumor ~ clump_thick + adhesion + mitoses + missing + nuclei,
    data = trainDummy, family = binomial(link = "logit"))
summary(model)
```

```
##
## Call:
## glm(formula = tumor ~ clump_thick + adhesion + mitoses + missing +
##     nuclei, family = binomial(link = "logit"), data = trainDummy)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.439  -0.197  -0.077   0.029   2.130
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.4813     0.9405   -9.02  < 2e-16 ***
## clump_thick   0.7311     0.1284    5.69  1.3e-08 ***
## adhesion      0.5265     0.1077    4.89  1.0e-06 ***
## mitoses       0.5200     0.2438    2.13    0.033 *
## missing1      0.8932     1.1302    0.79    0.429
## nuclei        0.5812     0.0922    6.31  2.8e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 641.03  on 493  degrees of freedom
## Residual deviance: 119.28  on 488  degrees of freedom
## AIC: 131.3
##
## Number of Fisher Scoring iterations: 8
```

```r
# accuracy and confusion matrix
predDummy <- predict(model, test, type = "response")
predDummy <- as.factor(ifelse(predDummy > 0.5, 1, 0))
mean(predDummy == test$tumor)
```

```
## [1] 0.966
```

```r
table(predDummy, test$tumor)
```

```
##
## predDummy   0   1
##         0 136   5
##         1   2  62
```