Appendix

Setup for producing the data frame

```r
# This script produces birth.data, which is the data to be used for
# the regression model, and train.inds, the indices for the train-set
# and num.train which is the size of the training set
# and MSPE function for calculating MSPE for a given model (uses train.ind)
births <- read.csv("chds_births.csv")
meth.names <- c('Caucasian','Caucasian','Caucasian','Caucasian',
'Caucasian','Caucasian','Mexican', 'African-American',
'Asian', 'Mixed', 'Other')
med.names <- c('elementary', 'middle', 'hs', 'hs + trade',
'hs + college', 'college', 'trade', 'unclear')
feth.names <- c('Caucasian','Caucasian','Caucasian','Caucasian',
'Caucasian','Cauasian', 'Mexican', 'African-American',
'Asian', 'Mixed', 'Other')
fed.names <- c('elementary', 'middle', 'hs', 'hs + trade',
'hs + college', 'college', 'trade', 'unclear')
marital.names <- c(NA, 'married', 'separated', 'divorced', 'widowed', 'never married')
income.names <- c('<2500', '2500-4999', '5000-7499', '7500-9999',
'10000-12499', '12500-14999', '15000-17499', '20000-22499', '>22500')
smoke.names <- c('never', 'now', 'until pregnancy', 'used to')
time.names <- c('never', 'still smokes', 'during pregnancy',
'less than a year', '1-2yrs', '2-3yrs', '3-4yrs', '5-9yrs',
'10+yrs', 'quit - unknown then')
number.names <- c('never', '1-4', '5-9', '10-14', '15-19',
'20-29', '30-39', '40-60', '>60', 'smoked, amount unknown')
births$meth <- meth.names[births$meth + 1]
births$feth<- feth.names[births$feth + 1]
births$fed <- fed.names[births$fed + 1]
births$marital <- marital.names[births$marital+1]
births$income <- income.names[fdata$income + 1]
births$smoke <- smoke.names[births$smoke + 1]
births$time <- time.names[births$time + 1]
births$number <- number.names[births$number + 1]
keeps <- c("wt", "gestation", "parity", "time", "number", "smoke", "mage",
"mwt", "mht", "meth","income")
cat.var <- c("smoke", "number", "time", "income")
birth.data <- births[keeps]
birth.data <- na.omit(birth.data)
#Initial Models
ntot <- dim(birth.data)[1]
ntrain <- 1000
train.ind <- c(NA)
for (c in cat.var) {
  train.ind.curr <- OPTALLOC(birth.data, c, "wt", ntrain/length(cat.var), 23430)
  train.ind <- unique(c(train.ind, train.ind.curr))
}
train.ind <- na.omit(train.ind)
num.train <- length(train.ind)
```

```r
MSPE <- function(M, train) {
  print(M$call)
  print(sum((birth.data$wt[-train] - predict(M, newdata = birth.data[-train,]))^2))
}
```

Missing Data and Imputation for income covariate

```r
#To run this code, we need the following libraries
#* mice
#* VIM
#Reading in the data
fdata <- read.csv("chds_births.csv")
head(fdata)

#Calculat the number of missing valuesin each column
na_count <- sapply(fdata, function(y) sum(length(which(is.na(y)))))
na.count <- data.frame(na_count)
na.count

#Check the columns that have more than 10% of the data missing
count <- sapply(fdata, function(y) length(y))
na_percent <- (na_count/count)*100
na_percent <- data.frame(na_percent)
na_percent

library(VIM)
```

```r
#Plot to visualize missing data
#miss_plot <- aggr(fdata, col=c('navyblue','yellow'),
  #                   numbers=TRUE, sortVars=TRUE,
  #                  labels=names(fdata), cex.axis=.7,
  #                gap=3, ylab=c("Missing data","Pattern"))

#Deleting father's weight and height due to missing data
fdata$fht <- NULL
fdata$fwt <- NULL
colnames(fdata)
head(fdata)

library(mice)
#The md.pattern function allows us to get a better understanding of the pattern of missing data
#md.pattern(fdata)
```

```r
#Imputting the data with "pmm" method as referenced from "https://stefvanbuuren.name/mice/"
imp <- mice(fdata, m = 5, maxit = 50, meth = 'pmm', seed = 500)

#This shows the imputation for each of the 5 iterations
head(complete(imp))
#Looking at one of the complete datasets #4
head(complete(imp,2))
summary(imp)
```

```r
#Plot of density functions of imputed data overlayed on the observed values for each of the data.
#densityplot(imp)
#blue - observed obseravtions
#magenta - imputed values

imp_1 <- data.frame(complete(imp,1))
imp_2 <- data.frame(complete(imp,2))
imp_3 <- data.frame(complete(imp,3))
imp_4 <- data.frame(complete(imp,4))
imp_5 <- data.frame(complete(imp,5))

cm1 <- sapply(fdata, mean, na.rm = T) - sapply(imp_1, mean)
cm2 <- sapply(fdata, mean, na.rm = T) - sapply(imp_2, mean)
cm3 <- sapply(fdata, mean, na.rm = T) - sapply(imp_3, mean)
cm4 <- sapply(fdata, mean, na.rm = T) - sapply(imp_4, mean)
cm5 <- sapply(fdata, mean, na.rm = T) - sapply(imp_5, mean)
sum_imp_data <- c(sum(cm1), sum(cm2), sum(cm3), sum(cm4), sum(cm5))
abs(sum_imp_data)

#We choose to include the imputed income data from imputed dataset 1 into our
#original dataset and then carry #our model diagnostics and selection from there.
#Get sample 1 income values
imputed_income <- imp_1$income
fdata$income <- imputed_income
```

Optimal Allocation

```r
OPTALLOC <- function(df, stratum, y, n, seed){
  #
  # OPTALLOC returns the training set selected via.
  # stratified random sampling with optimal allocation
  # for a given categorical variable "stratum"
  #   Input
  #     df: a dataframe
  #     stratum: the categorical variable whose levels
  #               are the strata
  #     y: the response covariate name
  #     n: the desired size of the training set
  #     seed: the seed for the random number generator
  #
  set.seed(seed) # set seed
  N <- length(df[,stratum]) #population size

  #initialize vectors
  vars <- c(rep (NA, length(unique(df[stratum]))))
  Wh <- c(rep(NA, length(unique(df[stratum]))))
  nh <- c(rep(NA, length(unique(df[stratum]))))
  counter <- 1

  #group by strata
  for (x in split(birth.data, birth.data[stratum])) {
    if (is.na(var(x[1][,y]))) {
      vars[counter] <- 0
    } else {
      vars[counter] <- var(x[1][,y])
    }
    Wh[counter] <- length(x[1][,y])/N
    nh[counter] <- sqrt(vars[counter])*(length(x[1][,y])/N)
    if(length(x[1]) > 0 && nh[counter] ==0 ){
      nh[counter] <- 1
    }
    counter <- counter + 1
  }

  #calculate nh
  den <- sum(nh)
  nh <- round(nh/den * n)

  #stratified sampling
  counter <- 1
  train.inds = c(NA)
  for (x in split(birth.data, birth.data[stratum])) {
    train.inds = c(train.inds, sample(as.numeric(rownames(x)), nh[counter]))
    counter <- counter + 1
  }

  return(train.inds)
}
```

Automatic Model Selection

```r
# Selection of Candidate Models (Automatic and Manual)

#source("setup.R")
set.seed(6024)
MSPE <- function(M, train) {
  print(M$call)
  print(sum((birth.data$wt[-train] - predict(M, newdata = birth.data[-train,]))^2))
}
```

```r
#Initial Models
M0 <- lm(wt ~ 1, data = birth.data, subset = train.ind)
Mmax <- lm(wt ~ gestation + parity + time + number+ smoke + mage + mwt + mht
           + I(mwt*703/(mht)^2) + I(gestation*parity) + I(gestation*mage)
           + income + meth, data=birth.data, subset = train.ind)
Mstart <- lm(wt ~ gestation + mage + I(mwt*703/(mht)^2) + smoke + mht + mwt
             + income, data=birth.data, subset = train.ind)
ntot <- dim(birth.data)[1]
ntrain <- num.train
# forward selection
Mfwd <- step(object = M0, # starting point model
scope = list(lower = M0, upper = Mmax), # smallest and largest model
direction = "forward",
trace = FALSE) # trace prints out information
# backward selection
Mback <- step(object = Mmax, # starting point model
scope = list(lower = M0, upper = Mmax),
direction = "backward", trace = FALSE)
# stepwise selection (both directions)
Mstep <- step(object = Mstart,
scope = list(lower = M0, upper = Mmax),
direction = "both", trace = FALSE)
```

```r
#MSPE(Mfwd, train.ind)
#MSPE(Mstep, train.ind)
#MSPE(Mback, train.ind)
```

```r
summary(Mstep)$coefficients[non.cat.inds,]
```

```r
#smoke
Mstep.smoke.red <- lm(formula = wt ~ mage + I(mwt * 703/(mht)^2)+ mht
                  + meth + I(gestation * mage) + I(gestation * parity),
                  data= birth.data, subset = train.ind)
```

```r
anova(Mstep.smoke.red, Mstep)
```

```r
#mother's ethnicity
Mstep.meth.red <- lm(formula = wt ~ mage + I(mwt * 703/(mht)^2)+ mht
                  + smoke + I(gestation * mage) + I(gestation * parity),
                  data= birth.data, subset = train.ind)
```

```r
anova(Mstep.meth.red, Mstep)
```

```r
Mstep.new <- lm(formula = wt ~ mage + I(mwt * 703/(mht)^2) + smoke + mht +
meth + I(gestation * mage) + I(gestation * parity) + parity, data = birth.data,
subset = train.ind)
non.cat.inds <- c(14, 15)
```

```r
summary(Mstep.new)$coefficients[non.cat.inds,]
```

```r
non.cat.inds <- c(1, 2, 3, 7, 8)
```

```r
M.manual <- lm(wt ~ gestation + mht + smoke + I(gestation*parity) + parity + number
               + income, data = birth.data, subset = train.ind)
summary(M.manual)$coefficients[non.cat.inds, ]
```

```r
#smoke
M.manual.smoke.red <-  lm(wt ~ gestation + mht  + I(gestation*parity) + parity +
number + income, data = birth.data, subset=train.ind)
```

```
anova(M.manual.smoke.red, M.manual)
```

```
#number
M.manual.number.red <-  lm(wt ~ gestation + mht +smoke + I(gestation*parity) + parity +
income, data = birth.data, subset=train.ind)
```

```
anova(M.manual.number.red, M.manual)
```

```
#income
M.manual.income.red <-  lm(wt ~ gestation + mht +smoke + I(gestation*parity) +
  parity + number, data = birth.data, subset=train.ind)
anova(M.manual.income.red, M.manual)
M.2 <- lm(wt ~ gestation + mht + smoke + I(gestation*parity) + parity, data = birth.data,
subset = train.ind)
summary(M.2)$coefficients[non.cat.inds,]
```

```
M.2.smoke.red <-  lm(wt ~ gestation + mht  + I(gestation*parity) + parity,
                     data = birth.data, subset=train.ind)
anova(M.2.smoke.red, M.2)
Mstep$call
M.2$call
```

Model Comparison

```
# Automatic
M.auto <- lm(formula = wt ~ mage + I(mwt * 703/(mht)^2) + smoke + mht + meth +
I(gestation * mage) + I(gestation * parity), data = birth.data,
subset = train.ind)
#Manual Selection
M.manual <- lm(formula = wt ~ gestation + mht + smoke + I(gestation * parity)
+ parity, data = birth.data, subset = train.ind)
zres <- residuals(M.auto)
sig.hat <- sqrt(sum(zres^2)/(length(zres)-2))
zres <- zres/sig.hat
```

Residual Plots

Model 1 (Selected via. Automated Model Selection)

```
#residuals vs. fitted values
par(mfrow = c(1,2))
#plot(predict(M.auto), residuals(M.auto), main="Residuals vs. Fitted Values")
#abline(h = 0, col = "red", lty = 2) #horizontal line
#standardize residuals
#qqnorm(zres, main = "QQ-Plot")
#qqline(zres, col='red', lty = 2)
```

Model 2 (Selected via. Automated Model Selection)

```
zres <- residuals(M.manual)
sig.hat <- sqrt(sum(zres^2)/(length(zres)-2))
zres <- zres/sig.hat
```

```
#residuals vs. fitted values
#par(mfrow = c(1,2))
#plot(predict(M.manual), residuals(M.manual), main="Residuals vs. Fitted Values")
#abline(h = 0, col = "red", lty = 2) #horizontal line
#standardize residuals
#qqnorm(zres, main = "QQ-Plot")
#qqline(zres, col='red', lty = 2)
```

K fold cross validation

```
set.seed(2)
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

```
#We use the caret library to perform K fold cross validation
#Create the indices for the k folds, with 2/3rd of the data #being used as the training set
ind= createDataPartition(birth.data$wt, p = 2/3, list = FALSE )
#Using the train.ind from the above function, we can calculate the training and testing set
trainDF <- birth.data[ind, ]
testDF <- birth.data[-ind, ]
#The ControlParameters specify the cross fold validation method for 5 folds
ControlParameters <- trainControl(method = "cv",
number = 10, savePredictions = TRUE, classProbs = TRUE)
#Cross Validation for the first model, which was selected through automatic model selection.
#Training the model on the training set using the specified control parameters
M.auto_train <- train(wt ~ parity + time + mage
+ mht + I(mwt * 703/(mht)^2) + I(gestation * mage) +  meth,
data = trainDF, method = "lm",trControl  = ControlParameters, na.action = na.omit)
#Predictions on the test data from the automatic model
M.auto_predictions <- predict(M.auto_train, testDF)
#Observed values of weight for the test data indices
observed_data_test <-  birth.data[-ind,]$wt
#Calculate the MPSE
M.auto_mspe <- sum((M.auto_predictions - observed_data_test)^2)
#For Manual model
#Cross Validation for the second model, which was manually selected
#We can use the same control parameters
#Training the model on the training set using the specified control parameters
M.manual_train <- train(wt ~ gestation + mht + smoke + I(gestation*parity) +
parity + number +income, data = trainDF,
```

```r
method = "lm",trControl  = ControlParameters, na.action = na.omit)
#Predictions on the test data from the manual data
M.manual_predictions <- predict(M.manual_train, testDF)
#Calculate the MPSE
M.manual_mspe <- sum((M.manual_predictions - observed_data_test)^2)
#Display nicely
signif(c(Auto_model = M.auto_mspe, Manual_model = M.manual_mspe))
```

Leverege and Influence measures

Model 1

```r
Leverage <- hat(model.matrix(M.auto))
h <- hatvalues(M.auto)
n<-nobs(M.auto)
#cook's distance vs. leverage
D <- cooks.distance(M.auto)
infl.ind <- which.max(D)
hbar <- length(coef(M.auto))/n
lev.ind <- h > 2*hbar
clrs <- rep("black", len=n)
clrs[lev.ind] <- "blue"
clrs[infl.ind] <- "red"
par(mfrow = c(1, 1))
cex <- .8
```

```r
#plot(h, D, xlab = "Leverage", ylab="Cook's Influence Measure", pch=21, bg=clrs,
 #    cex=cex, cex.axis = cex)
#abline(v=2*hbar, col="grey", lty=2)
#legend("topleft", legend=c("High Leverage", "High Influence"), pch = 21,
#pt.bg = c("blue", "red"), cex=cex, pt.cex = cex)
```

```r
print(birth.data[infl.ind,])
```

Model 2

```r
Leverage <- hat(model.matrix(M.manual))
h <- hatvalues(M.manual)
n<-nobs(M.manual)
#cook's distance vs. leverage
D <- cooks.distance(M.manual)
infl.ind <- which.max(D)
hbar <- length(coef(M.manual))/n
lev.ind <- h > 2*hbar
clrs <- rep("black", len=n)
clrs[lev.ind] <- "blue"
clrs[infl.ind] <- "red"
par(mfrow = c(1, 1))
cex <- .8
```

```
#plot(h, D, xlab = "Leverage", ylab="Cook's Influence Measure", pch=21, bg=clrs, #cex=cex, cex.axis = c
#abline(v=2*hbar, col="grey", lty=2)
```

```
print(birth.data[infl.ind,])
```

AIC

```
# models to compare
M1 <- M.auto
M2 <- M.manual
AIC(M1)
AIC(M2)
```

PRESS STATISTIC

```
# models to compare
M1 <- M.auto
M2 <- M.manual
# PRESS statistics
press1 <- resid(M1)/(1-hatvalues(M1)) # M1
press2 <- resid(M2)/(1-hatvalues(M2)) # M2
# plot PRESS statistics
#boxplot(x = list(abs(press1), abs(press2)), names = c("Automatic", "Manual")
#,ylab = expression(group("|", PRESS[i], "|")),
#col = c("yellow", "orange"))
```