1. Summary

The objective of this report is to build a prediction model for the neurological behavior of children aged 6-11 who participated in the HELIX study. *REDACTED*

2. Exploratory Data Analysis

Data Cleaning

```
# indices of the units where our outcome of interest in NA
NA_idx <- which(is.na(phenotypeNA$hs_Gen_Tot))
# Complete case analysis
# sorted by ID and with NAs removed
outcomes <- phenotypeNA[order(phenotypeNA$ID),][-NA_idx,]
exposome <- exposomeNA[order(exposomeNA$ID),][-NA_idx,] # 223 covariates in total
paste("There are", length(colnames(exposome)), "covariates to start with.")
```

```
## [1] "There are 223 covariates to start with."
```

```
num.NA <- length(NA_idx)
prop <- num.NA/length(phenotypeNA$hs_Gen_Tot)
paste("There are", num.NA, "data values missing in the outcome of interest.")
```

```
## [1] "There are 8 data values missing in the outcome of interest."
```

```
paste("This is only", round(prop*100, 4), "% of the units.")
```

```
## [1] "This is only 0.6149 % of the units."
```

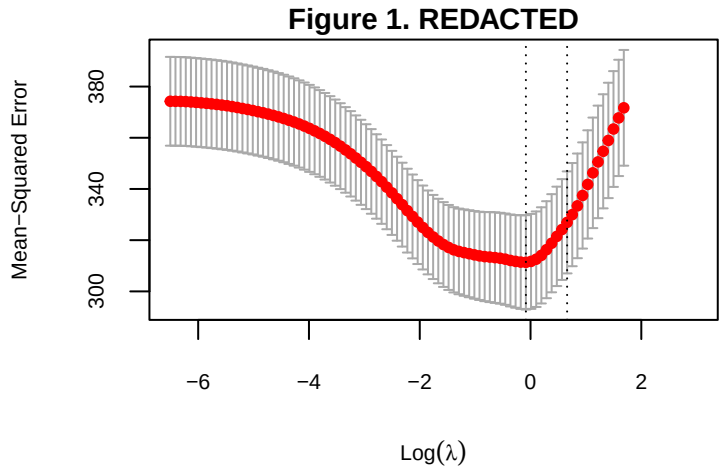
In terms of a better method, **multiple imputation** can be considered when there is a large proportion of missing data.

Test & Train Data

This means 90% of the data is for training and the other 10% is used for testing. *REDACTED*

```
set.seed(1) # reproducibility
complete.data <- cbind(exposome, data.frame(hs_Gen_Tot=outcomes$hs_Gen_Tot))
complete.data = complete.data[, which(colnames(complete.data) != "ID")]
len <- length(outcomes$hs_Gen_Tot)
# choose 129 observations as test data (about 10%) and others are training data
ntot <- nrow(complete.data)
ind <- sample(len, size = ceiling(ntot*0.1))
chosen_rows <- row.names(exposome)[ind]
test.data = complete.data[(rownames(complete.data) %in% chosen_rows), ]
training.data <- complete.data[!(rownames(complete.data) %in% chosen_rows), ]
```

```
suppressPackageStartupMessages(suppressWarnings(library(glmnet)))
set.seed(1) # reproducibility
L1.matrix <- model.matrix(~.-hs_Gen_Tot, data=training.data)[, -1]
L1.lassocv <- cv.glmnet(x=L1.matrix, y=training.data$hs_Gen_Tot, alpha=1)
par(mar=c(4.1,4.1,1.1,1.1))
plot(L1.lassocv, xlim=c(-6.5, 3), main="Figure 1. REDACTED", cex = 1, cex.lab = 0.7, cex.main = 0.9, ce
```



Having performed LASSO with cross-validation on the training data, we see from the plot that a reasonable range of $\log(\lambda)$ is around $(-0.3, 0.6)$, and we will proceed first by choosing the λ values that gives the minimum mean-squared errors.

Level 1: lambda.min In using the lambda value which gives the absolute minimzed MSE:

```
L1.coef <- coef(L1.lassocv, s="lambda.min")
# extract the names of the covariates that are bigger than 0
L1.covleft <- which(L1.coef[, 1] != 0)
paste("There are", length(L1.covleft)-1, "covariates left. ")
```

```
## [1] "There are 35 covariates left. "
```

```
L1.coefmag <- L1.coef[as.numeric(L1.covleft)]
L1.coefnames <- names(L1.covleft)
# a dataframe that reports the name of the covaraites left and their magnitude
L1.coefdf <- data.frame("names" = L1.coefnames,
"magnitude" = L1.coefmag)
L1.coefdf <- L1.coefdf[-1,]
```

```
L1.pred <- predict(L1.lassocv, newx=model.matrix(~.-hs_Gen_Tot ,data = test.data)[, -1],
s="lambda.min")
```

```
mean((L1.pred - test.data$hs_Gen_Tot)^2)
```

```
## [1] 270.2231
```

```
suppressPackageStartupMessages(suppressWarnings(library("sjmisc")))
fix.category <- function(x, df) {
i <- which(str_contains(x, colnames(df), ignore.case =
F))
colnames(df)[i]
}
M1.covfix <- fix.category(L1.coefdf$names, df = training.data) # fix categorical cov
# fitting the lm() mdoel
M1.covfix <- unique(M1.covfix)
M1.training = training.data[, which(colnames(training.data) %in% c(M1.covfix, "hs_Gen_Tot"))]
M1 <- lm(hs_Gen_Tot ~ ., data = M1.training)
```

Level 2: A large lambda By choosing a really large lambda, we know the prediction model directly coming from LASSO with cross-validation is highly biased, but we will have a clear idea of what important covariates we need to include to fit a small `lm` model. *REDACTED*

```
lambda_2 = 2 # a really large lambda choice
M2.coef <- coef(L1.lassocv, s=lambda_2)
M2.covleft <- which(M2.coef[, 1] != 0)
paste("There are", length(M2.covleft)-1, "covariates left. ")
```

```
## [1] "There are 15 covariates left. "
```

```
M2.coefmag <- M2.coef[as.numeric(M2.covleft)]
M2.coefnames <- names(M2.covleft)
# a dataframe that reports the name of the covaraites left and their magnitude
M2.coefdf <- data.frame("names" = M2.coefnames,
"magnitude" = M2.coefmag)
M2.coefdf <- M2.coefdf[-1,]
```

```
M2.covfix <- fix.category(M2.coefdf$names, df = training.data) # fix categorical cov
# fitting the lm() mdoel
M2.covfix <- unique(M2.covfix)
M2.training = training.data[, which(colnames(training.data) %in% c(M2.covfix, "hs_Gen_Tot"))]
M2 <- lm(hs_Gen_Tot ~ ., data = M2.training)
```

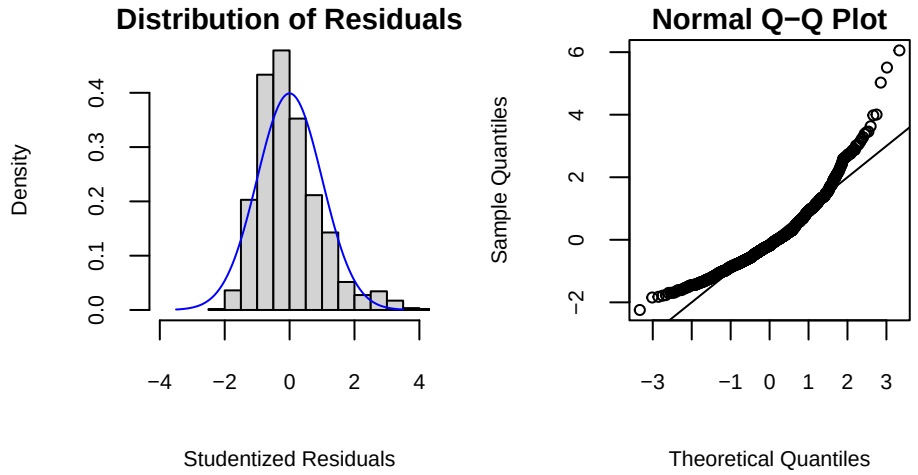
Model Assumptions

Since our objective is to build prediction models, we need that the four assumptions about multiple linear regression models are met. We proceed by checking the LINE assumptions of `L1`.

Normality

```
# residuals
res <- resid(M2) # raw residuals
stud <- res/(sigma(M2)*sqrt(1-hatvalues(M2))) # studentized residuals
par(mfrow=c(1, 2), mar=c(4.1,4.1,1.1,1.1))
## plot distribution of studentized residuals
```

```
hist(stud,breaks=25,
probability=TRUE,xlim=c(-4,4),
xlab="Studentized Residuals",
main="Distribution of Residuals",
cex = 1, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
grid <- seq(-3.5,3.5,by=0.05)
lines(x=grid,y=dnorm(grid),col="blue") # add N(0,1) pdf
## qqplot of studentized residuals
qqnorm(stud, cex = 0.7, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
abline(0,1) # add 45 degree line
```

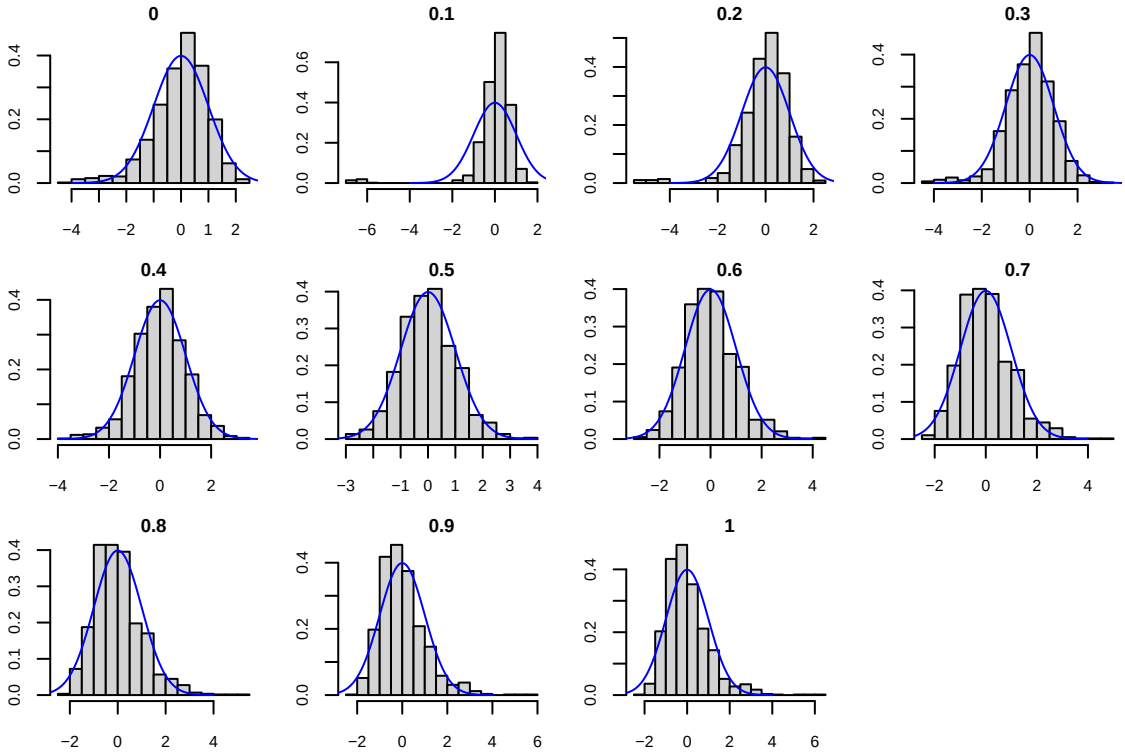


Plotting a histogram and qq-plot of the studentized residuals reveals that normality assumption is not met. The histogram is rather right-skewed, and the qqplot has a quadratic shape.

We proceed by applying power transformations on the outcome to fix the problem.

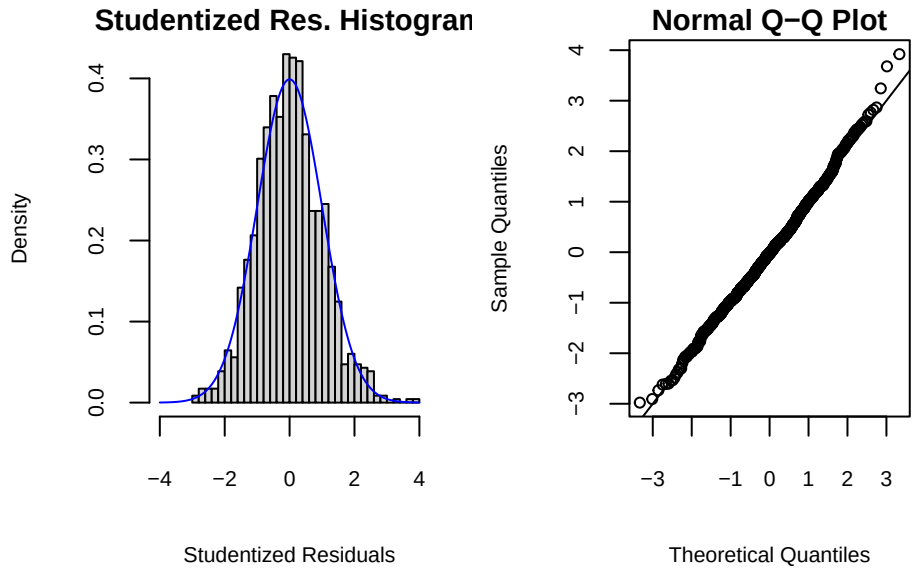
```
powerfun = function(x, alpha) {
if (alpha == 0) log(x+1) # add one to avoid Inf
else if (alpha > 0) {
x^alpha
} else -x^alpha
}
powers <- seq(0, 1, by=0.1)
par(mfrow=c(3,4), mar=c(2.6,2.1,1.1,1.1))
for(ii in 1:length(powers)) {
Mtemp.training = training.data[, which(colnames(training.data) %in% c(M2.covfix, "hs_Gen_Tot"))]
Mtemp.training$hs_Gen_Tot = powerfun(Mtemp.training$hs_Gen_Tot, powers[ii])
M3 <- lm(hs_Gen_Tot ~ ., data = Mtemp.training)
Mtemp <- lm(hs_Gen_Tot ~ ., data = Mtemp.training)
## residuals
res <- resid(Mtemp) # raw residuals
stud <- res/(sigma(Mtemp)*sqrt(1-hatvalues(Mtemp))) # studentized residuals
hist(stud,breaks = 15,
probability=TRUE,
xlab="", ylab="",
main=powers[ii],
cex = 1, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
grid <- seq(-4,4,by=0.05)
```

```
lines(x=grid,y=dnorm(grid),col="blue") # add N(0,1) pdf
}
```



The power $\alpha = 0.5$ looks best so we will proceed by using the square root on the outcomes.

```
M3.training = training.data[, which(colnames(training.data) %in% c(M2.covfix, "hs_Gen_Tot"))]
M3.training$hs_Gen_Tot = sqrt(M3.training$hs_Gen_Tot) # sqrt transform
names(M3.training)[names(M3.training) ==
'hs_Gen_Tot'] <- 'sqrt_hs_Gen_Tot'
M3 <- lm(sqrt_hs_Gen_Tot ~ ., data = M3.training)
## residuals
res <- resid(M3) # raw residuals
stud <- res/(sigma(M3)*sqrt(1-hatvalues(M3))) # studentized residuals
par(mfrow=c(1,2), mar=c(4.1,4.1,1.1,1.1))
# plot distribution of studentized residuals
hist(stud,breaks = 25,
probability = TRUE,
xlim=c(-4, 4),
xlab="Studentized Residuals",
main = "Studentized Res. Histogram",
cex = 1, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
grid <- seq(-4,4,by=0.05)
lines(x=grid,y=dnorm(grid),col="blue") # add N(0,1) pdf
# qqplot of studentized residuals
qqnorm(stud, cex = 0.7, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
abline(0,1) # 45 degree line
```



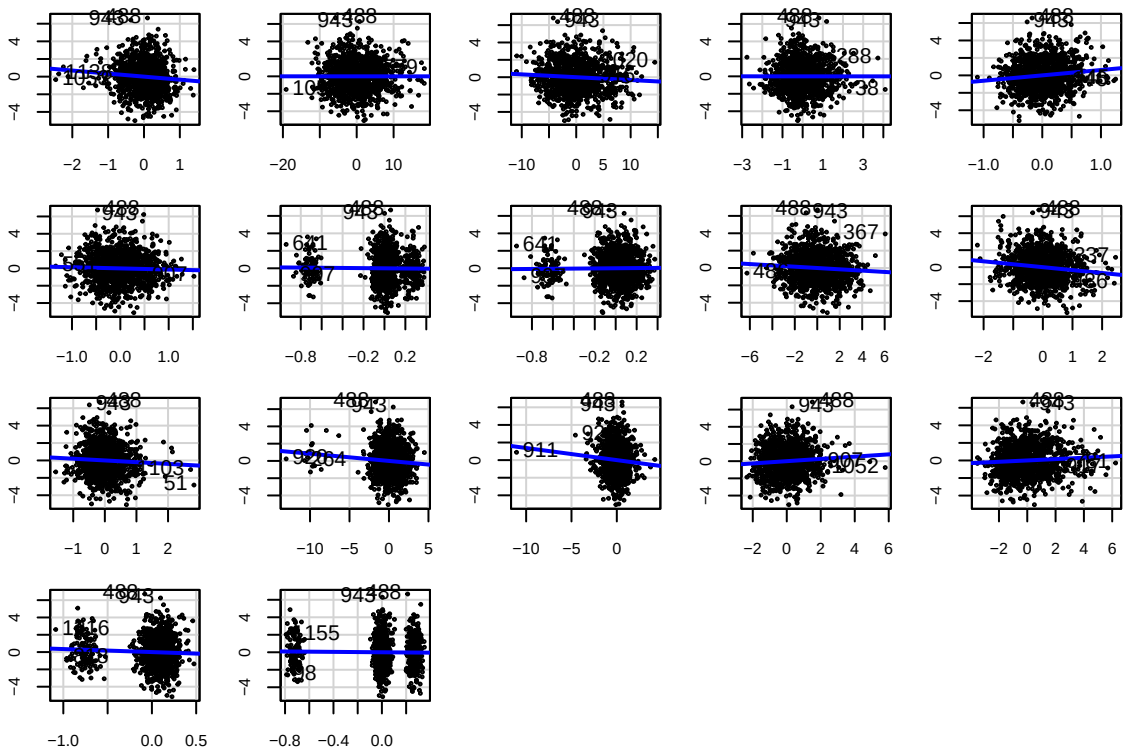
Clearly, the histogram and the qqplot look much better.

Independence

REDACTED

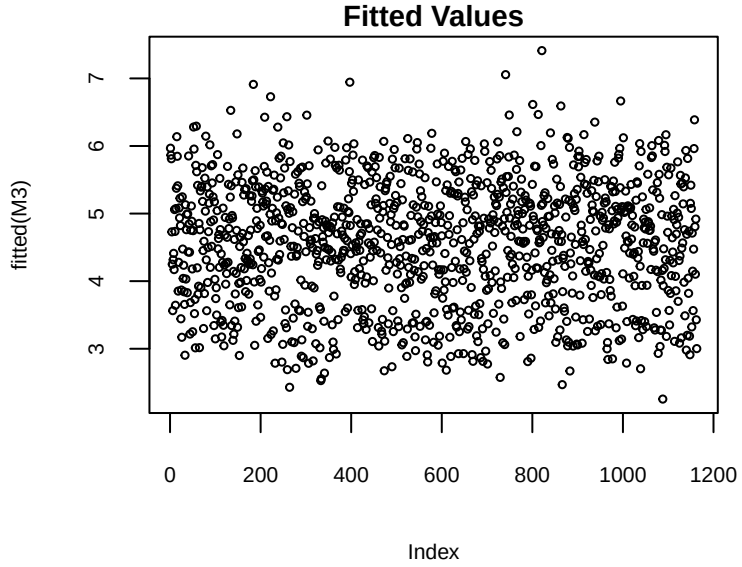
Linearity

```
suppressPackageStartupMessages(suppressWarnings(library(car)))
par(mfrow=c(4,5), mar=c(2.1,2.1,1.1,1.1))
avPlots(M3, layout=NA, ask = FALSE,
xlab = "", ylab = "",
cex = 0.3, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
```



Note there is really no prospect for a categorical covariate to be nonlinear. It's domain of values are discrete and quantized such that no trend can be discerned.

```
# plot of studentized residuals vs fitted values
par(mfrow=c(1,1), mar=c(4.1,4.1,1.1,1.1))
plot(fitted(M3), main = "Fitted Values",
cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
```

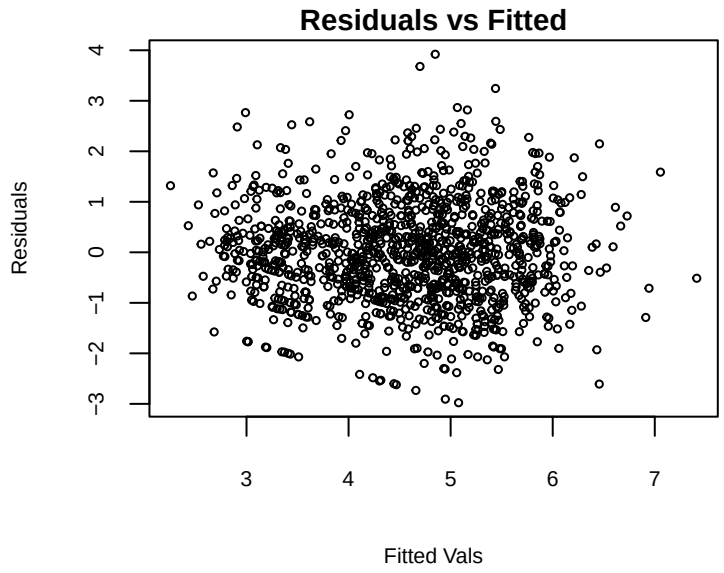


This plot also suggests that we do not have a reasonable basis to add higher order terms, such as quadratic terms, in our model.

We conclude that we the linearity assumption is met reasonably well.

Homoskedasticity

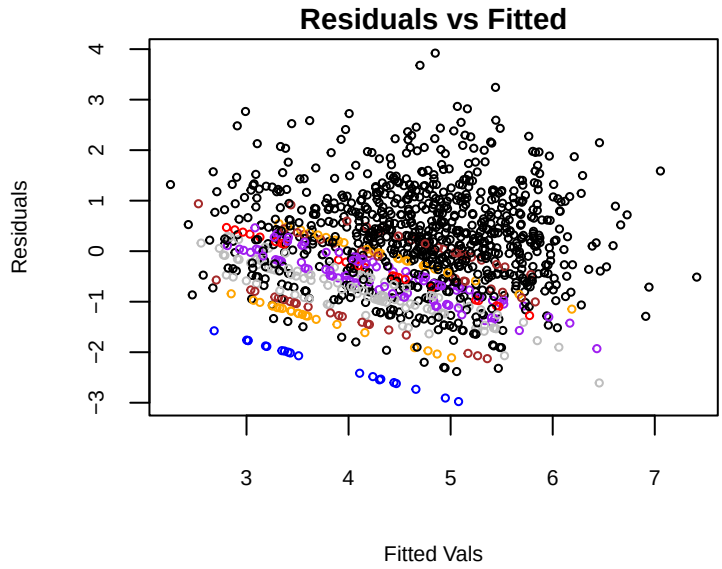
```
par(mfrow=c(1,1), mar=c(4.1,4.1,1.1,1.1))
plot(stud~fitted(M3),
xlab="Fitted Vals",
ylab="Residuals",
main="Residuals vs Fitted",
cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
```



From the plot above, we seem to be meeting the **homoskedasticity** assumption. Notice that there are some suspicious linear trends on the lower left half of the plot. These trendlines are likely due to the outcome being discrete, we can confirm this by investigating statistical modes:

```
Mode <- function(x) {
m <- na.omit(unique(x))
m[which.max(tabulate(match(x, m)))]
}
iterations <- 15
col.opts <- c("red", "blue", "green", "brown", "orange", "purple", "grey")
colours <- rep("black", nrow(M3.training))
mode.vector <- M3.training$sqrt_hs_Gen_Tot
mode.outcomes <- Mode(mode.vector)
mode.idx <- which(mode.vector == mode.outcomes)
colours[mode.idx] <- sample(col.opts, 1)
for (i in 2:iterations) {
mode.vector[mode.idx] <- NA
mode.outcomes <- Mode(mode.vector)
mode.idx <- which(mode.vector == mode.outcomes)
colours[mode.idx] <- sample(col.opts, 1)
}
par(mfrow=c(1,1), mar=c(4.1,4.1,1.1,1.1))
```

```
plot(stud~fitted(M3),
xlab="Fitted Vals",
ylab="Residuals",
main="Residuals vs Fitted",
col=colours,
cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
```



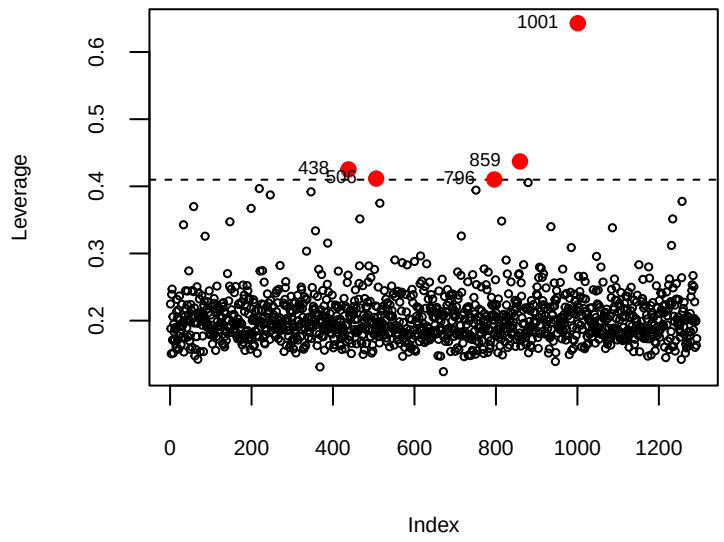
After identifying the hierarchical statistical modes and colouring each value-wise, it appears that the linear trends correspond to these statistical modes in the outcome, which occur due to the outcome being discrete. It is also generally the case that most participants score relatively low on the CBCL scale, and hence many attain identical scores.

Outliers

Leverage

REDACTED

```
suppressPackageStartupMessages(suppressWarnings(library(readr)))
# compute leverage
Mfull <- lm(hs_Gen_Tot ~ ., data = complete.data)
Xmat <- model.matrix(Mfull) ## design matrix
H <- Xmat%*%solve(t(Xmat)%*%Xmat)%*%t(Xmat) ## Hat matrix
lev <- hatvalues(Mfull) ## leverage (h_i)
hbar <- mean(lev) ## \bar{h}
# plot leverage
par(mfrow=c(1,1), mar=c(4.1,4.1,1.1,1.1))
plot(lev,ylab="Leverage", cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
abline(h=2*hbar,lty=2) ## add line at 2hbar
ids <- which(lev>2*hbar) ## x values for labelling points >2hbar
points(lev[ids]~ids,col="red",pch=19) ## add red points >2hbar
text(x=ids,y=lev[ids], labels=ids, cex= 0.6, pos=2) ## label points >2hbar
```



The highlighted red points are those leverages which are larger than the mean value of the diagonals of the hat matrix H.

```
## investigate high leverage points
largest.coef <- which(round(abs(Mfull$coefficients[-1]),1) == round(max(abs(Mfull$coefficients[-1])), 1
largest.cov <- names(largest.coef)
paste("The strongest covariate in the model by coefficient is", largest.cov)
```

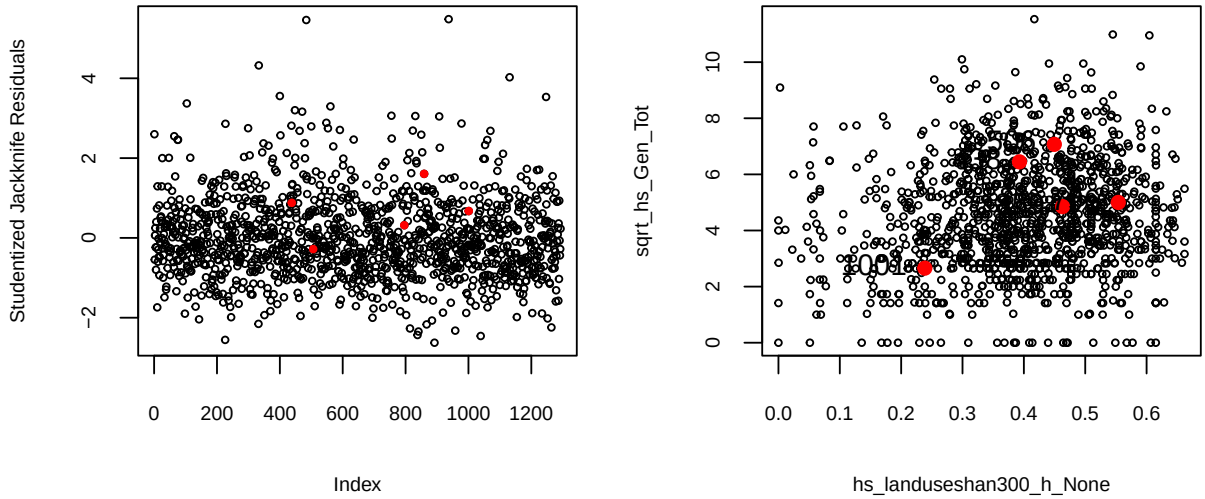
```
## [1] "The strongest covariate in the model by coefficient is hs_landuseshan300_h_None"
```

Jackknife Resiudals

```
jack3 <- rstudent(Mfull)

par(mfrow=c(1,2), mar=c(4.1,4.1,1.1,1.1))
plot(jack3,ylab="Studentized Jackknife Residuals",
     cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
points(jack3[ids]~ids,col="red",pch=19,
       cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7) # add high leverage points

plot(sqrt(complete.data$hs_Gen_Tot)~complete.data$hs_landuseshan300_h_None,ylab="sqrt_hs_Gen_Tot",xlab=
     cex = 0.5, cex.lab = 0.7, cex.main = 0.9, cex.axis = 0.7)
points(sqrt(complete.data$hs_Gen_Tot)[ids]~complete.data$hs_landuseshan300_h_None[ids],col="red",pch=19
text(sqrt(complete.data$hs_Gen_Tot)[ids]~complete.data$hs_landuseshan300_h_None[ids], labels=ids, cex=
```



We do not believe it is appropriate here to exclude outliers as all the values fall in a valid range of the CBCL scoring mechanism, so there is no evidence of data entry errors. Additionally we should be particularly interested in units that scored high on the scale as they exhibit extreme neurological behaviour.

Interactions

Level 1: lambda.min We proceed by investigating possible interactions in `M1`.

```
M1.covfix <- fix.category(L1.coefdf$names, df = training.data) # fix categorical cov
M1.training = training.data[, which(colnames(training.data) %in% c(M1.covfix, "hs_Gen_Tot"))]
M1.training$hs_Gen_Tot = sqrt(M1.training$hs_Gen_Tot)
names(M1.training)[names(M1.training) ==
'hs_Gen_Tot'] <- 'sqrt_hs_Gen_Tot'
M1 <- lm(sqrt_hs_Gen_Tot ~ ., data = M1.training)
```

```
M1.coef <- coef(M1)
M1.names <- fix.category(names(M1.coef)[2:length(M1.coef)], M1.training)
M1.names <- unique(M1.names)
M1.names
```

```
##  [1] "hs_no2_yr_hs_h_Log"          "h_connind300_preg_Sqrt"
##  [3] "h_fdensity300_preg_Log"      "hs_accesslines300_s_dic0"
##  [5] "h_Benzene_Log"               "h_PM_Log"
##  [7] "h_TEX_Log"                   "e3_alcpreg_yn_None"
##  [9] "h_fastfood_preg_Ter"         "hs_bakery_prod_Ter"
## [11] "hs_dairy_Ter"                "hs_KIDMED_None"
## [13] "hs_pet_cat_r2_None"          "hs_sd_wk_None"
## [15] "hs_dif_hours_total_None"     "hs_cs_m_Log2"
## [17] "hs_cu_c_Log2"                "hs_cu_m_Log2"
## [19] "hs_mo_m_Log2"                "hs_pb_c_Log2"
## [21] "hs_tl_mdich_None"            "hs_ndvi100_h_None"
## [23] "hs_pcb180_cadj_Log2"         "hs_dep_cadj_Log2"
## [25] "hs_pfoa_c_Log2"              "hs_pfoa_m_Log2"
## [27] "hs_pfos_c_Log2"              "hs_mepa_madj_Log2"
## [29] "hs_mbzp_cadj_Log2"           "hs_mecpp_cadj_Log2"
```

```
## [31] "hs_meohp_cadj_Log2"          "hs_mep_cadj_Log2"
## [33] "FAS_cat_None"                "hs_participation_3cat_None"
## [35] "hs_smk_parents_None"
```

From our research, these are the interactions we consider: *REDACTED* Considering all possible combinations of interactions, we have $2^6 - 1 = 63$ possibilities.

```
interactions.names <- c("hs_pcb180_cadj_Log2 : h_Benzene_Log", "hs_mo_m_Log2 : h_Benzene_Log", "hs_pfos

# to obtain the multisets of the interactions we will consider
multisets <- function(x) {
  m <- lapply(1:length(x), function(y) {
    combn(x, y)
  })
  return(m)
}

# application of multisets on interaction names
inter.combn <- multisets(interactions.names)

# multisets of sizes 1 through 6
inter1 <- inter.combn[[1]]
inter2 <- inter.combn[[2]]
inter3 <- inter.combn[[3]]
inter4 <- inter.combn[[4]]
inter5 <- inter.combn[[5]]
inter6 <- inter.combn[[6]]

# function to obtain models from multisets
get.interaction.model <- function(terms, covname, model.training) {
  sapply(1:ncol(terms), function(x) {
    lm(as.formula(paste("sqrt_hs_Gen_Tot", "~",
                paste(covname, collapse = "+"), "+",
                paste(terms[,x], collapse = "+"),
                          sep = "")), data = model.training)

  })
}
```

```
# models using interactions from multisets of sizes 1 through 6
M.inter1 <- get.interaction.model(inter1,  M1.covfix, M1.training)
M.inter2 <- get.interaction.model(inter2,  M1.covfix, M1.training)
M.inter3 <- get.interaction.model(inter3,  M1.covfix, M1.training)
M.inter4 <- get.interaction.model(inter4,  M1.covfix, M1.training)
M.inter5 <- get.interaction.model(inter5,  M1.covfix, M1.training)
M.inter6 <- get.interaction.model(inter6,  M1.covfix, M1.training)
```

```
M1.vif <- vif(M1)
M1vif.fixname <- fix.category(rownames(M1.vif), df=training.data)
tolerence = 2
```

```
highvif.ind <- c()
for (i in 1:length(M1.vif)) {
  if (rownames(M1.vif)[i] %in% M1vif.fixname & abs(M1.vif[i]) >= 2) {
    highvif.ind <- append(highvif.ind, i)
  }
}
M1.vif[highvif.ind]
```

Colinearity

```
## [1] 2.572644 2.309783 2.023330 3.036628 3.061737
```

We removed the terms with high VIF and fit the model again.

```
highvif <- rownames(M1.vif)[highvif.ind]
M1vif.leftname <- M1vif.fixname[-which(M1vif.fixname %in% highvif)]
M1new.training <- M1.training[, which(colnames(M1.training) %in% c(M1vif.leftname, "sqrt_hs_Gen_Tot"))]
M1.new <- lm(sqrt_hs_Gen_Tot ~ ., data = M1new.training)
```

After removing terms with high VIF, we refit the model with interaction terms.

```
# NEW CHUNK
M1new.coef <- coef(M1.new)
M1new.names <- fix.category(names(M1new.coef)[2:length(M1new.coef)], M1new.training)
M1new.names <- unique(M1new.names)
M1new.names
```

```
##  [1] "h_connind300_preg_Sqrt"     "hs_accesslines300_s_dic0"
##  [3] "h_Benzene_Log"              "h_PM_Log"
##  [5] "h_TEX_Log"                  "e3_alcpreg_yn_None"
##  [7] "h_fastfood_preg_Ter"        "hs_bakery_prod_Ter"
##  [9] "hs_dairy_Ter"               "hs_KIDMED_None"
## [11] "hs_pet_cat_r2_None"         "hs_sd_wk_None"
## [13] "hs_dif_hours_total_None"    "hs_cs_m_Log2"
## [15] "hs_cu_c_Log2"               "hs_cu_m_Log2"
## [17] "hs_mo_m_Log2"               "hs_pb_c_Log2"
## [19] "hs_tl_mdich_None"           "hs_pcb180_cadj_Log2"
## [21] "hs_dep_cadj_Log2"           "hs_pfoa_c_Log2"
## [23] "hs_pfoa_m_Log2"             "hs_pfos_c_Log2"
## [25] "hs_mepa_madj_Log2"          "hs_mbzp_cadj_Log2"
## [27] "hs_mep_cadj_Log2"           "FAS_cat_None"
## [29] "hs_participation_3cat_None" "hs_smk_parents_None"
```

```
# NEW CHUNK
interactions.names <- c("hs_pcb180_cadj_Log2 : h_Benzene_Log", "hs_mo_m_Log2 : h_Benzene_Log", "hs_pfos

# application of multisets on interaction names
inter.combn <- multisets(interactions.names)

# multisets of sizes 1 through
inter1 <- inter.combn[[1]]
inter2 <- inter.combn[[2]]
```

```
inter3 <- inter.combn[[3]]
inter4 <- inter.combn[[4]]
inter5 <- inter.combn[[5]]
```

```
c(inter1, inter2, inter3, inter4, inter5)
```

```
##  [1] "hs_pcb180_cadj_Log2 : h_Benzene_Log" "hs_mo_m_Log2 : h_Benzene_Log"
##  [3] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_cs_m_Log2  : e3_alcpreg_yn_None"
##  [5] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
##  [7] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pcb180_cadj_Log2 : h_Benzene_Log"
##  [9] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [11] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [13] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_mo_m_Log2 : h_Benzene_Log"
## [15] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_mo_m_Log2 : h_Benzene_Log"
## [17] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_mo_m_Log2 : h_Benzene_Log"
## [19] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pfos_c_Log2 : h_Benzene_Log"
## [21] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_pfos_c_Log2 : h_Benzene_Log"
## [23] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [25] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [27] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pfos_c_Log2 : h_Benzene_Log"
## [29] "hs_pcb180_cadj_Log2 : h_Benzene_Log" "hs_mo_m_Log2 : h_Benzene_Log"
## [31] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [33] "hs_mo_m_Log2 : h_Benzene_Log"        "e3_alcpreg_yn_None :  hs_cu_m_Log2"
## [35] "hs_pcb180_cadj_Log2 : h_Benzene_Log" "hs_pfos_c_Log2 : h_Benzene_Log"
## [37] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [39] "hs_pfos_c_Log2 : h_Benzene_Log"      "e3_alcpreg_yn_None :  hs_cu_m_Log2"
## [41] "hs_pcb180_cadj_Log2 : h_Benzene_Log" "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [43] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_mo_m_Log2 : h_Benzene_Log"
## [45] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [47] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pfos_c_Log2 : h_Benzene_Log"
## [49] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_mo_m_Log2 : h_Benzene_Log"
## [51] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "e3_alcpreg_yn_None :  hs_cu_m_Log2"
## [53] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [55] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [57] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pfos_c_Log2 : h_Benzene_Log"
## [59] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [61] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pfos_c_Log2 : h_Benzene_Log"
## [63] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [65] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [67] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [69] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [71] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_mo_m_Log2 : h_Benzene_Log"
## [73] "hs_pfos_c_Log2 : h_Benzene_Log"      "hs_cs_m_Log2  : e3_alcpreg_yn_None"
## [75] "e3_alcpreg_yn_None :  hs_cu_m_Log2"  "hs_pcb180_cadj_Log2 : h_Benzene_Log"
## [77] "hs_mo_m_Log2 : h_Benzene_Log"        "hs_pfos_c_Log2 : h_Benzene_Log"
## [79] "hs_cs_m_Log2  : e3_alcpreg_yn_None"  "e3_alcpreg_yn_None :  hs_cu_m_Log2"
```

Level 2: A large lambda Switching back to `M3`:

```
M3.coef <- coef(M3)
M3.names <- fix.category(names(M3.coef)[2:length(M3.coef)], M3.training)
M3.names <- unique(M3.names)
```

By observing the terms left, we need to decide the interaction terms that are sensible to add. *REDACTED* Considering all existing scenarios, we have $2^3 - 1 = 7$ possibilities.

```
# Fit the model with interaction terms as discussed above
M4 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2, data = M3.training)
M5 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_smk_parents_None, data = M3.training)
M6 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pfos_c_Log2, data = M3.training)
M7 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_smk_parents_None,
        data = M3.training)
M8 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_pfos_c_Log2,
        data = M3.training)
M9 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_smk_parents_None+h_Benzene_Log:hs_pfos_c_Log2,
        data = M3.training)
M10 <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_smk_parents_None+
           h_Benzene_Log:hs_pfos_c_Log2,data = M3.training)
```

Colinearity After transforming our `lm` model to meet the assumptions and investigating the possible interaction terms, we start to examine the collinearity in `M3`. *REDACTED*

```
suppressPackageStartupMessages(suppressWarnings(library(DAAG)))
vif(M3)
```

```
##             hs_no2_yr_hs_h_Log           hs_pm10_yr_hs_h_None
##                         1.8967                         2.4076
##         h_connind300_preg_Sqrt         h_fdensity300_preg_Log
##                         1.5121                         2.3330
##                 h_Benzene_Log                      h_TEX_Log
##                         1.2689                         1.5229
## h_fastfood_preg_Ter(0.25,0.83]  h_fastfood_preg_Ter(0.83,Inf]
##                         4.8998                         5.6400
##                hs_KIDMED_None       hs_dif_hours_total_None
##                         1.1916                         1.2119
##                  hs_cs_m_Log2           hs_pcb180_cadj_Log2
##                         1.4328                         1.2651
##                hs_pfos_c_Log2            hs_mecpp_cadj_Log2
##                         1.3121                         1.3177
##               hs_mep_cadj_Log2    hs_smk_parents_Noneneither
##                         1.1678                         3.0269
##         hs_smk_parents_Noneone
##                         2.7135
```

REDACTED

```
M3new.training <- M3.training[, which(colnames(M3.training) != "hs_pm10_yr_hs_h_None")]
M3.new <- lm(sqrt_hs_Gen_Tot ~ ., data = M3new.training)
```

Since our interaction terms do not include `hs_pm10_yr_hs_h_None`, we can build new models with interaction as below.

```
M4.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2, data = M3new.training)
M5.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_smk_parents_None, data = M3new.training)
M6.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pfos_c_Log2, data = M3new.training)
```

```
M7.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_smk_parents_None,
        data = M3new.training)
M8.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_pfos_c_Log2,
        data = M3new.training)
M9.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_smk_parents_None+h_Benzene_Log:hs_pfos_c_Log2,
        data = M3new.training)
M10.new <- lm(sqrt_hs_Gen_Tot ~ .+h_Benzene_Log:hs_pcb180_cadj_Log2+h_Benzene_Log:hs_smk_parents_None+
           h_Benzene_Log:hs_pfos_c_Log2,data = M3new.training)
```

Level 3: A small lambda (about 100-120 covariates) This level exists purely for the sake of comparison. We know if we fit a full model,which is a model that includes all the covariates (223 covariates in total), it is not likely to be a good prediction model. However, we do not know how many covariates the best prediction will have, so we need to examine all levels and keep them in our pool for safety.

```
lambda_3 <- 0.3 #L3
M11.coef <- coef(L1.lassocv, s=lambda_3)
# extract the names of the covariates that are bigger than 0
M11.covleft <- which(M11.coef[, 1] != 0)
paste("There are", length(M11.covleft)-1, "covariates left. ")
```

```
## [1] "There are 139 covariates left. "
```

```
M11.coefmag <- M11.coef[as.numeric(M11.covleft)]
M11.coefnames <- names(M11.covleft)
# a dataframe that reports the name of the covaraites left and their magnitude
M11.coefdf <- data.frame("names" = M11.coefnames,
                         "magnitude" = M11.coefmag)

M11.covfix <- fix.category(M11.coefdf$names, df = training.data) # fix categorical cov
# fitting the lm() mdoel
M11.training = training.data[, which(colnames(training.data) %in% c(M11.covfix, "hs_Gen_Tot"))]
M11.training$hs_Gen_Tot = sqrt(M11.training$hs_Gen_Tot) # sqrt root transform
M11 <- lm(hs_Gen_Tot ~ ., data = M11.training)
```

```
# function to obtain models from multisets
get.interaction.model <- function(terms, covname, model.training) {
  sapply(1:ncol(terms), function(x) {
    lm(as.formula(paste("hs_Gen_Tot", "~",
                paste(covname, collapse = "+"), "+",
                paste(terms[,x], collapse = "+"),
                          sep = "")), data = model.training)
  })
}

# Add the same inetraction terms as before
M11.inter1 <- get.interaction.model(inter1,  M11.covfix, M11.training)
M11.inter2 <- get.interaction.model(inter2,  M11.covfix, M11.training)
M11.inter3 <- get.interaction.model(inter3,  M11.covfix, M11.training)
M11.inter4 <- get.interaction.model(inter4,  M11.covfix, M11.training)
M11.inter5 <- get.interaction.model(inter5,  M11.covfix, M11.training)
M11.inter6 <- get.interaction.model(inter6,  M11.covfix, M11.training)
```

Level 4: a lambda that gives around 80 covairates There is a huge fluctuation on lambda.min. *REDACTED*

```
lambda_4 <- 0.55 #L3
M12.coef <- coef(L1.lassocv, s=lambda_4)
# extract the names of the covariates that are bigger than 0
M12.covleft <- which(M12.coef[, 1] != 0)
paste("There are", length(M12.covleft)-1, "covariates left. ")
```

```
## [1] "There are 77 covariates left. "
```

```
M12.coefmag <- M12.coef[as.numeric(M12.covleft)]
M12.coefnames <- names(M12.covleft)
# a dataframe that reports the name of the covaraites left and their magnitude
M12.coefdf <- data.frame("names" = M12.coefnames,
                         "magnitude" = M12.coefmag)

M12.covfix <- fix.category(M12.coefdf$names, df = training.data) # fix categorical cov
# fitting the lm() mdoel
M12.training = training.data[, which(colnames(training.data) %in% c(M12.covfix, "hs_Gen_Tot"))]
M12.training$hs_Gen_Tot = sqrt(M12.training$hs_Gen_Tot) # sqrt trasnform
M12 <- lm(hs_Gen_Tot ~ ., data = M11.training)
```

REDACTED

```
# Add the same inetraction terms as before
M12.inter1 <- get.interaction.model(inter1,  M12.covfix, M12.training)
M12.inter2 <- get.interaction.model(inter2,  M12.covfix, M12.training)
M12.inter3 <- get.interaction.model(inter3,  M12.covfix, M12.training)
M12.inter4 <- get.interaction.model(inter4,  M12.covfix, M12.training)
M12.inter5 <- get.interaction.model(inter5,  M12.covfix, M12.training)
M12.inter6 <- get.interaction.model(inter6,  M12.covfix, M12.training)
```

Grouped LASSO

Notice that so far all the models we have done derives from LASSO with cross-validation, however the process does not treat categorical covariates in the best way. *REDACTED* Note that if we apply group LASSO, the issue we saw before will be solved since all the levels of a categorical covariates will shrink together.

```
suppressPackageStartupMessages(suppressWarnings(library(gglasso)))
get.numlevels <- function(dat, catnames) {
  vec <- 1:length(colnames(dat))
  cnames <- colnames(dat)
  for (i in 1:length(cnames)) {
    if (cnames[i] %in% catnames) {
      level <- unique(dat[, i])
      vec[i] <- length(level)
    } else {
      vec[i] <- 1
    }
  }
  return(vec)
```

```
}

all_names <- colnames(training.data)
all_ind <- 1:length(all_names)
category.codebook <- codebook[which(codebook$var_type == "factor"), ]
category.name <- rownames(category.codebook)
numlevels <- get.numlevels(training.data[, colnames(training.data) != "hs_Gen_Tot" ], category.name)
```

```
get.group <- function(numlevel) {
  groupnum <- c()
  for (i in 1:length(numlevel)) {
    if (numlevel[i] == 1) {
      one_group <- rep(i, each=numlevel[i])
      groupnum <- append(groupnum, one_group)
    } else {
      one_group <- rep(i, each=numlevel[i]-1)
      groupnum <- append(groupnum, one_group)
    }
  }
  return(groupnum)
}

groupnum <- get.group(numlevels)
```

```
trial <- gglasso(x=L1.matrix, y=training.data$hs_Gen_Tot, group = groupnum, lambda=2)
c <- predict(trial, newx=model.matrix(~.-hs_Gen_Tot ,data = test.data)[, -1],
        s="lambda.1se")
```

```
mean((c - test.data$hs_Gen_Tot)^2)
```

```
## [1] 259.6013
```