Vision Project Part 2

Brian Dang & Ethan Mai & Naomi Wilcox

06 February 2023

Table of Contents: 1. Normalize variables via z-score for kNN 2. Create training and testing sets 3. kNN: Test Initial Models 4. kNN: Determine Optimal k 5. kNN: Final Model 6. Multinomial Regression 7. Pairs plots 8. Lasso 9. Finding optimal lambda 10. Lasso Prediction

Normalize variables via z-score for kNN

```
summary(data$srf3)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000117 0.001325 0.067572 0.016275 4.516317
```

```
summary(data$shrm3) #these two are similar in mean
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0003013 0.0058991 0.0938365 0.0601057 2.9447333
```

```
summary(data$irf3) #very small mean
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000075 0.0001140 0.0135362 0.0012087 1.3047842
```

```
summary(data$ped3) #both large in mean
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00035  0.08447  0.20831  0.48968  0.52487 10.40473
```

```
summary(data$rpe3) #though RPE has a small maximum value
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4415  0.7336  0.7802  0.7833  0.8322  1.0753
```

```
#boxplot()
```

We conduct kNN with the Euclidean distance metric. As suggested by the above, the effect in kNN of one biomarker is generally not equal to those of the others. For instance, IRF biomarker is in a significantly smaller numeric scale than other variables. Normalization improves this to generally give equal weight to each predictor. Here we standardize by the z-score: http://cs.wellesley.edu/~cs305/lectures/3_kNN.pdf

Trade-off here: the Euclidean distances in the kNN used are no longer indicative of the volume. Despite this we care more about each feature having equal weight.

```
#Gender made binary for kNN
data$gender.n=as.numeric(data$gender)
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$gender)){
  if (data$gender[i] == "Male") {
    data$gender.n[i] = -1
  } else {
    data$gender.n[i] = 1
  }
}

##Ethnicity for kNN
data$eth.afro=as.numeric(data$ethnicity) #initialize
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$ethnicity)){
  if (data$ethnicity[i] == "afrocaribbean") {
    data$eth.afro[i] = 1
  } else {
    data$eth.afro[i] = -1
  }
}

data$eth.asian=as.numeric(data$ethnicity) #initialize
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$ethnicity)){
  if (data$ethnicity[i] == "asian") {
    data$eth.asian[i] = 1
  } else {
    data$eth.asian[i] = -1
  }
}

data$eth.cau=as.numeric(data$ethnicity) #initialize
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$ethnicity)){
  if (data$ethnicity[i] == "caucasian") {
    data$eth.cau[i] = 1
  } else {
    data$eth.cau[i] = -1
  }
}

data$eth.other=as.numeric(data$ethnicity) #initialize
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$ethnicity)){
  if (data$ethnicity[i] == "Other") {
    data$eth.other[i] = 1
  } else {
    data$eth.other[i] = -1
  }
}
```

```
data$eth.unknown=as.numeric(data$ethnicity) #initialize
```

```
## Warning: NAs introduced by coercion
```

```
for (i in 1:length(data$ethnicity)){
  if (data$ethnicity[i] == "unknown") {
    data$eth.unknown[i] = 1
  } else {
    data$eth.unknown[i] = -1
  }
}

#levels(as.factor(data$agegroup))
levels(as.factor(data$ethnicity))
```

```
## [1] "afrocaribbean" "asian"         "caucasian"     "Other"
## [5] "unknown"
```

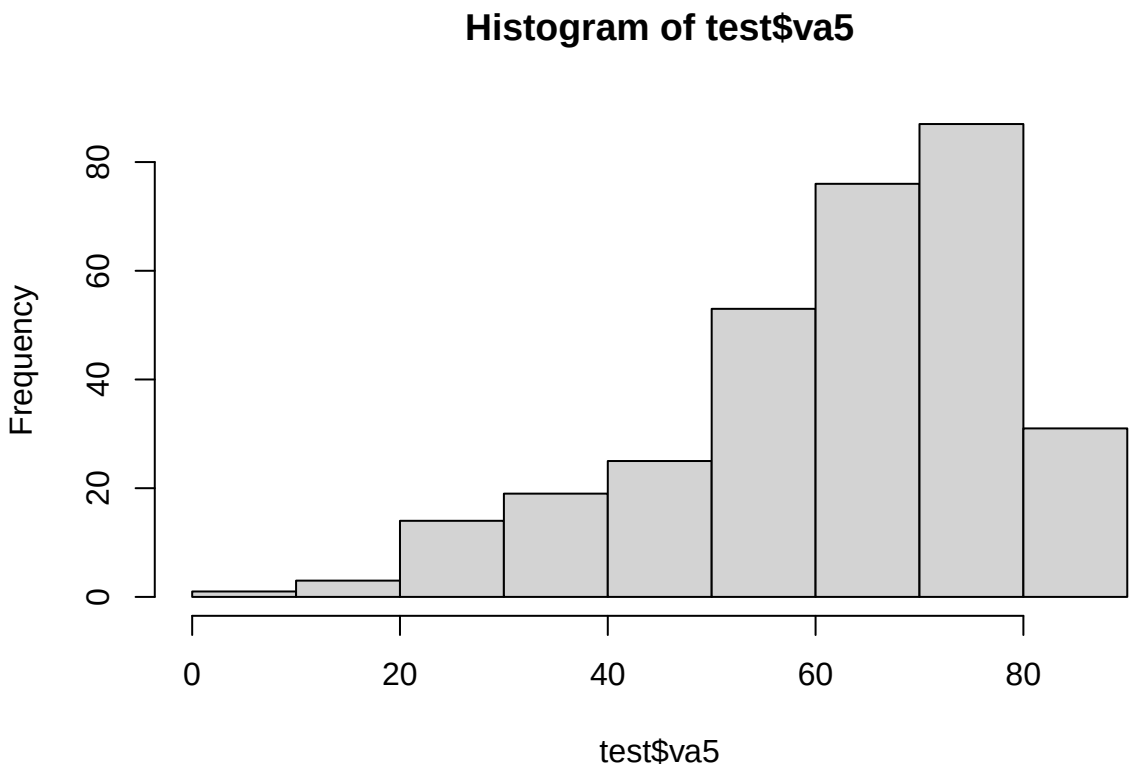
```
#standardization of biomarkers for kNN
colnames(data[,c(13, 18, 23, 28, 33, 8)])
```

```
## [1] "irf3"  "rpe3"  "srf3"  "ped3"  "shrm3" "va3"
```

```
indx=c(13, 18, 23, 28, 33, 8)
for (i in 1:6){
  data[,indx[i]]=(data[,indx[i]]-mean(data[,indx[i]]))/sd(data[,indx[i]]) #standardize using z-score
}

#summary(data[,13]) #double check
#summary(data[,18])
```

Create training and testing sets

```
nn <- dim(data)[1]
set.seed(123)
tst <- sample(1:nn, ceiling(nn/3), replace = F)
test <- data[tst,]
train <- data[-tst,]

#Comparing the two sets
hist(test$va5)
```




```
hist(train$va5)
```


```
table(test$class)
```

```
##
##   1   2   3   4   5
```

```
##  30  85  76 112   6
```

```
table(train$class)
```

```
##
##   1   2   3   4   5
##  53 190 159 202  13
```

kNN: Test Initial Models

The non-numerical features used here are: 'agegroup', 'gender', and 'ethnicity'. We treat these as follows:

-Age group is ordinal. There does not seem to be an easy way to conform this to the Euclidean distance metric, so we omit this variable. -Gender is binary, coded as -1 for male, 1 for female. This coding is done to make it similar to the z-score normalized data used in the other predictors. -Ethnicity is made into a binary variable, in the same manner as gender, for each measured ethnicity. Therefore there are 5 binary dummy variables to encompass the following ethnicity categories in the data: Afro-Caribbean, Asian, Caucasian, Other, and Unknown.

We attempt the following combinations of predictors: only biomarkers+VA, all predictors, biomarkers+VA+gender, and biomarkers+VA+ethnicity.

```
ks=c(5, 9, 11, 13, 15, 21, 31) # choices of k that we will try
misclass.knn=as.data.frame(cbind(ks, ks, ks, ks)) #initialize
misclass.knncv=as.data.frame(cbind(ks, ks)) #initialize

rownames(misclass.knn)=c("k=5", "9", "11", "13", "15", "21", "31")
colnames(misclass.knn)=c("biomarker+VA misclassification", "full kNN", "
                         biomarkers+VA+gender", "biomarkers+VA+ethnicity")

xnames.bio <- c("ped3", "shrm3", "irf3", "srf3", "rpe3", "va3")
xnames <- c("ped3", "shrm3", "irf3", "srf3", "rpe3", "gender.n", "eth.afro",
            "eth.asian", "eth.cau", "eth.other", "eth.unknown", "va3")
xnames.gen <- c("ped3", "shrm3", "irf3", "srf3", "rpe3", "gender.n", "va3")
xnames.eth <- c("ped3", "shrm3", "irf3", "srf3", "rpe3", "eth.afro",
            "eth.asian", "eth.cau", "eth.other", "eth.unknown", "va3")

for (i in 1:length(ks)){
  knn.pred.bio <- knn(train[,xnames.bio], test[,xnames.bio],train$class, k=ks[i]) #biomarker+VA-only mo
  knn.pred <- knn(train[,xnames], test[,xnames],train$class, k=ks[i])
  #model with all predictors
  knn.pred.gen <- knn(train[,xnames.gen], test[,xnames.gen],train$class, k=ks[i])
  knn.pred.eth <- knn(train[,xnames.eth], test[,xnames.eth],train$class, k=ks[i])

  tbl.bio <- table(knn.pred.bio, test$class) #tables of predicted vs true class.
  tbl <- table(knn.pred, test$class) #diagonals are agreement b/t model and truth
  tbl.gen <- table(knn.pred.gen, test$class)
  tbl.eth <- table(knn.pred.eth, test$class)

  # (number of misclassifications)/total
  mis.bio=(nrow(test)-sum(diag(tbl.bio)))/nrow(test) # biomarker-only
  mis=(nrow(test)-sum(diag(tbl)))/nrow(test) #misclassification from full kNN
  mis.gen=(nrow(test)-sum(diag(tbl.gen)))/nrow(test)
  mis.eth=(nrow(test)-sum(diag(tbl.eth)))/nrow(test)

  misclass.knn[i,1]=mis.bio #store values
```

```
  misclass.knn[i,2]=mis
  misclass.knn[i,3]=mis.gen
  misclass.knn[i,4]=mis.eth
}
##k=5

misclass.knn
```

```
##     biomarker+VA misclassification  full kNN
## k=5                       0.5016181 0.5210356
## 9                         0.4433657 0.5177994
## 11                        0.4660194 0.4983819
## 13                        0.4530744 0.5177994
## 15                        0.4627832 0.5113269
## 21                        0.4530744 0.5210356
## 31                        0.4757282 0.5210356
##     \n                          biomarkers+VA+gender biomarkers+VA+ethnicity
## k=5                                        0.4951456               0.5145631
## 9                                          0.4983819               0.4789644
## 11                                         0.4886731               0.4757282
## 13                                         0.4951456               0.4789644
## 15                                         0.4789644               0.5080906
## 21                                         0.4822006               0.5113269
## 31                                         0.4757282               0.5307443
```

From the above we notice that the biomarker+VA misclassification is consistently the lowest, and thus the best. Thus, we continue with only the biomarkers+VA at time 3 as features.

kNN: Determine Optimal k

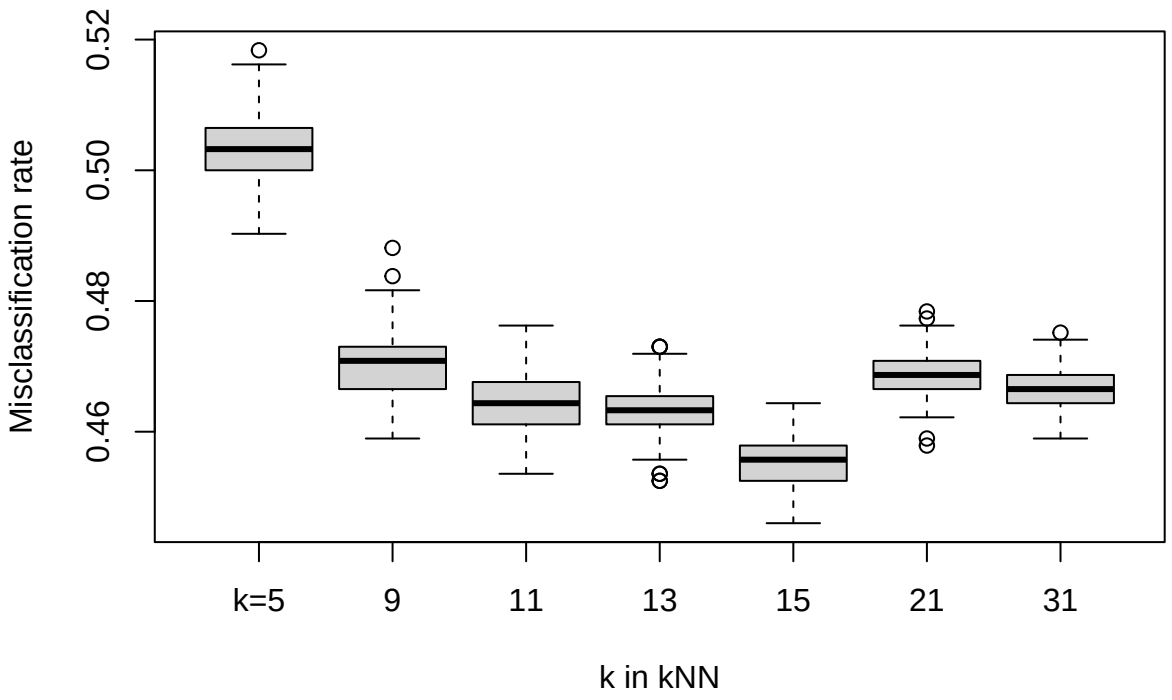
Below is repeated LOOCV with the entire dataset, not with the previously specified training set. LOOCV is implemented only for the purpose of finding the optimal k here.

```
##repeated LOOCV
library(boot)
numcv=200 #number of repetitions
cv.rep.mis <- as.data.frame(matrix(0, nrow=numcv, ncol=length(ks))) #dataframe of misclassification err
colnames(cv.rep.mis)=rownames(misclass.knn)

for (z in 1:numcv){
  for (i in 1:length(ks)){
    knncv.pred.bio <- knn.cv(data[,xnames.bio], data$class, k=ks[i])
    tbl.bio <- table(knncv.pred.bio, data$class)
    mis.bio=(nrow(data)-sum(diag(tbl.bio)))/nrow(data) # biomarker-only
    cv.rep.mis[z, i]=mis.bio #store values
  }
}
head(cv.rep.mis)
```

```
##         k=5         9        11        13        15        21        31
## 1 0.5053996 0.4730022 0.4589633 0.4665227 0.4546436 0.4697624 0.4676026
## 2 0.5075594 0.4730022 0.4676026 0.4697624 0.4589633 0.4686825 0.4665227
## 3 0.5032397 0.4708423 0.4676026 0.4643629 0.4503240 0.4697624 0.4643629
## 4 0.5032397 0.4589633 0.4643629 0.4632829 0.4568035 0.4686825 0.4676026
## 5 0.4978402 0.4643629 0.4751620 0.4578834 0.4524838 0.4676026 0.4643629
```

```
## 6 0.4946004 0.4719222 0.4643629 0.4622030 0.4568035 0.4708423 0.4654428
```

```
boxplot(cv.rep.mis, ylab='Misclassification rate', xlab='k in kNN',
        title='LOOCV over 200 repetitions')
```



kNN: Final Model

It seems that k=15 is optimal. Thus this is the k for our final model. Below we find the error rate over 2000 repetitions of this final model, given the original test-train data split.

```
mis.final=numeric(2000) # vector of misclassification error

for (z in 1:2000){
    knn.final <- knn(train[,xnames.bio], test[,xnames.bio],train$class, k=15) #biomarker+VA-only model
    tbl.bio <- table(knn.final, test$class)
    mis.bio=(nrow(test)-sum(diag(tbl.bio)))/nrow(test)
    mis.final[z]=mis.bio #store values
  }

summary(mis.final)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4401  0.4563  0.4628  0.4617  0.4660  0.4822
```

Multinomial Regression

```
data = read.csv('~/Downloads/dataframev2.csv')
data <- data %>%
  rename(
    "irf4" = "vol_irf4",
    "rpe4" = "vol_rpe4",
```

```
    "srf4" = "vol_srf4",
    "ped4" = "vol_ped4",
    "shrm4" = "vol_shrm4",
    "irhr4" = "vol_intrarethyperreflect4",
    "irf3" = "vol_irf3",
    "rpe3" = "vol_rpe3",
    "srf3" = "vol_srf3",
    "ped3" = "vol_ped3",
    "shrm3" = "vol_shrm3",
    "irhr3" = "vol_intrarethyperreflect3"
  )


data$agegroup <- factor(data$agegroup)
data$ethnicity <- factor(data$ethnicity)
data$gender <- factor(data$gender)
levels(data$gender) <- c(1,0) #female = 1
levels(data$agegroup) <- c(1,2,3,4) #1= 50-59,2= 60-69,3= 70-79,4= 80+
levels(data$ethnicity) <- c(1,2,3,4,5) #1=afrocaribbean, 2=asian, 3=caucasian, 4=other, 5=unkown

#Create new multiclass outcome variable
data$class <- cut(data$va5, breaks = c(100,85,70,60,35,0), labels = c(1,2,3,4,5))
data$class3 <- cut(data$va3, breaks = c(100,85,70,60,35,0), labels = c(1,2,3,4,5), include.lowest = T)
table(data$class)
```

```
##
##   1   2   3   4   5
##  83 275 235 314  19
```

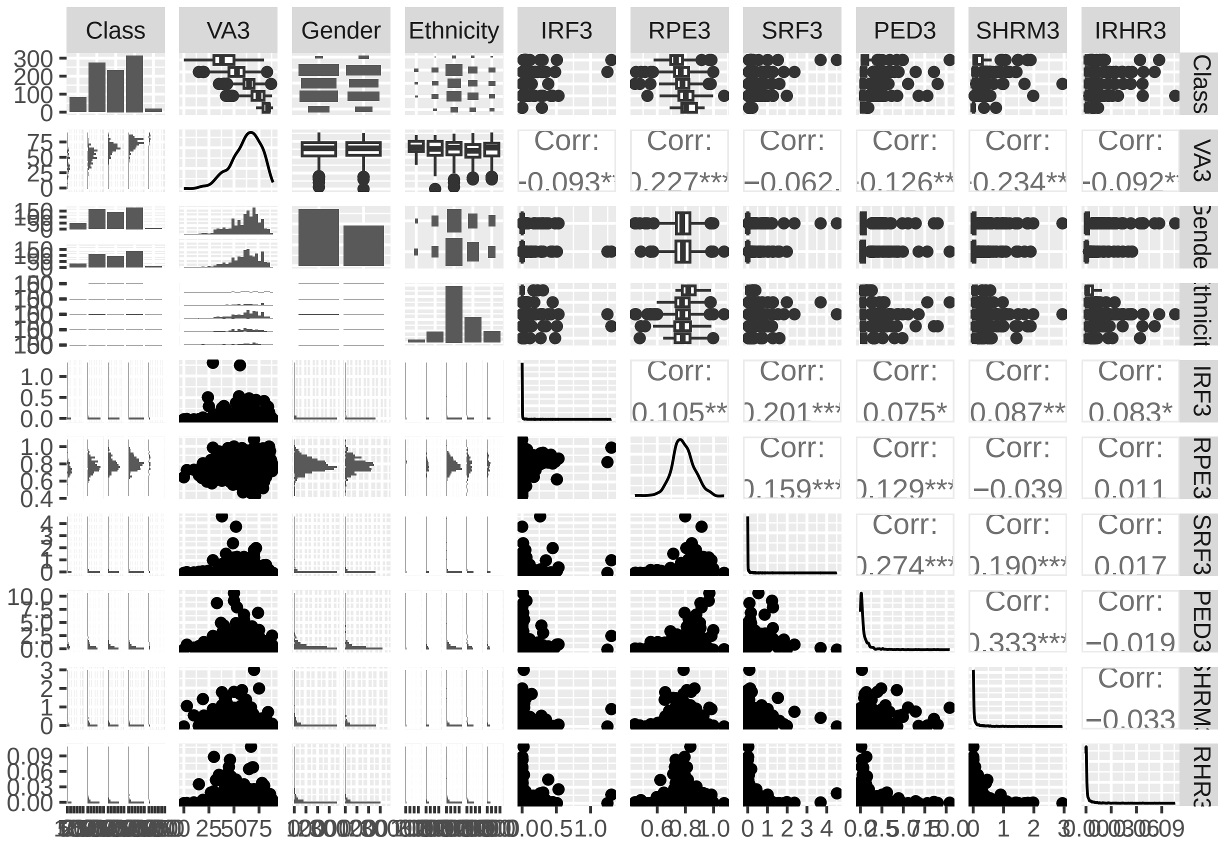
```
#1 = no visual impairement (VI), 2= mild VI, 3=moderate VI, 4=blind, 5=severe blindness


#create change in va1 and va3 variable
data$change <- cut(data$va3-data$va1, breaks = c(-Inf,0,Inf), labels = c("Decrease","Increase"))
data$blind <- ifelse(data$va5 < 60,1,0)
data$improve <- ifelse(data$va5 >= data$va3,1,0) #or time 1


#Create training and testing sets
nn <- dim(data)[1]
set.seed(123)
tst <- sample(1:nn, ceiling(nn/5), replace = F)
test <- data[tst,]
train <- data[-tst,]
```

Pairs plots

```
ggpairs(data, columns = c(42, 8,5,3, 13,18,23,28,33,38), columnLabels = c("Class","VA3","Gender","Ethni
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
#"changeirf","changerpe","changesrf","changeped","changeshrm","changeirhr"
xnams <- c("va3", "irf3", "rpe3", "srf3", "ped3", "shrm3","irhr3")
fmla2 <- as.formula(paste("class ~ ", paste(xnams, collapse= "+")))


mlogist <- multinom(fmla2, data=train, na.actoin=na.omit)
```

```
## # weights:  45 (32 variable)
## initial  value 1190.984055
## iter  10 value 861.339395
```

```
## iter  20 value 718.096548
## iter  30 value 696.554798
## iter  40 value 695.682928
## iter  50 value 695.626540
## iter  60 value 695.568077
## final  value 695.547020
## converged
```

```
smlogist <- summary(mlogist)
lab <- c("Incercept", "VA3","IRF3","RPE3","SRF3","PED3","SHRM3","IRHR3")
kable(smlogist$coefficients, caption = "Coefficients for multinomial regression", col.names = lab) %>%
  kable_styling(latex_options = c("striped")) %>%
  landscape
```

Table 1: Coefficients for multinomial regression

	Incercept	VA3	IRF3	RPE3	SRF3	PED3	SHRM3	IRHR3
2	-5.796935	0.0705481	-2.935630	5.436081	-0.4946993	-0.1636906	-0.7932920	-0.0092326
3	-14.454722	0.1764601	-5.650574	8.540481	-0.9057097	-0.1703629	-0.8891639	-18.0260271
4	-25.376145	0.2963765	-2.660174	12.088995	-1.1452596	-0.0208843	-1.6395916	-3.5782948
5	-52.671029	0.6259678	-1.839243	9.963135	-0.6394094	-0.5905356	0.5180646	-1.9749963

```
kable(smlogist$standard.errors, caption = "Standard errors for model coefficients", col.names = lab) %>%
  kable_styling(latex_options = c("striped")) %>%
  landscape
```

Table 2: Standard errors for model coefficients

	Incercept	VA3	IRF3	RPE3	SRF3	PED3	SHRM3	IRHR3
2	1.606364	0.0126963	2.577407	2.126717	0.3722240	0.1391251	0.4575732	7.0247001
3	1.894602	0.0169999	3.818871	2.287032	0.5834332	0.1734765	0.5519875	1.1778352
4	2.138464	0.0205627	3.524417	2.407747	0.6727531	0.1972782	0.7349145	3.4191307
5	2.953456	0.0661466	5.595544	3.686831	2.6280524	0.7231463	1.0306855	0.1921212

```r
#exp(coef(mlogist))

##inference
z <- summary(mlogist)$coeff/summary(mlogist)$standard.error
p <- (1-pnorm(abs(z),0,1))*2
kable(p, caption = "2-tailed z test for model coefficients", col.names = lab) %>%
  kable_styling(latex_options = c("striped")) %>%
  landscape
```

Table 3: 2-tailed z test for model coefficients

	Incercept	VA3	IRF3	RPE3	SRF3	PED3	SHRM3	IRHR3
2	0.0003077	0	0.2547089	0.0105856	0.1838359	0.2393667	0.0829724	0.9989513
3	0.0000000	0	0.1389679	0.0001882	0.1205714	0.3260742	0.1072144	0.0000000
4	0.0000000	0	0.4503784	0.0000005	0.0886902	0.9156918	0.0256814	0.2953067
5	0.0000000	0	0.7423842	0.0068850	0.8077718	0.4141457	0.6152169	0.0000000

Table 4: Test Set: Predicted vs. True classification

Predicted Class	True Class				
	1	2	3	4	5
1	3	4	0	0	0
2	10	35	11	8	0
3	2	10	18	14	0
4	0	4	14	48	5
5	0	0	0	0	0

Table 5: Training Set: Predicted vs. True classification

Predicted Class	True Class				
	1	2	3	4	5
1	23	13	0	0	0
2	42	154	53	13	0
3	2	32	69	35	0
4	1	23	70	196	11
5	0	0	0	0	3

```
kable(cbind(1:5,table(Predicted=predict(mlogist, newdata = test),True=test$class)), row.names = F, capti
  add_header_above(header = c(" ","True Class"=5))
```

```
kable(cbind(1:5,table(Predicted=predict(mlogist, newdata = train),True=train$class)), row.names = F, cap
  add_header_above(header = c(" ","True Class"=5))
```

```
##accuracy
sum(diag(table(predict(mlogist,newdata = test),test$class)))/sum(table(predict(mlogist, newdata = test)
```

```
## [1] 0.5591398
```

```
sum(diag(table(predict(mlogist,newdata = train),train$class)))/sum(table(predict(mlogist, newdata = trai
```

```
## [1] 0.6013514
```

$$p_I = P(Y = I|\vec{X}) = \frac{exp\left(\beta_0 + \beta_{1I}VA3 + \beta_{2I}IRF3 + \beta_{3I}RPE3 + ... + \beta_{8I}IRHR3\right)}{1 + \sum_{k=1}^{4} exp\left(\beta_0 + \beta_{1k}VA3 + \beta_{2k}IRF3 + \beta_{3k}RPE3 + ... + \beta_{8k}IRHR3\right)}$$

$$p_1 = P(Y = 1|\vec{X}) = \frac{1}{1 + \sum_{k=1}^{4} exp\left(\beta_0 + \beta_{1k}VA3 + \beta_{2k}IRF3 + \beta_{3k}RPE3 + ... + \beta_{8k}IRHR3\right)}$$

Lasso

```
xxx <- model.matrix(~.,train[,xnams])
yyy <- train$class
gridd <- c(exp(seq(2,-8,-0.5)),0)
lasso2 <- glmnet(xxx,yyy, family = "multinomial", alpha = 1, lambda = gridd)
lasso2$beta
```

```
## $`1`
## 8 x 22 sparse Matrix of class "dgCMatrix"
```

```
##    [[ suppressing 22 column names 's0', 's1', 's2' ... ]]
```

```
##
## (Intercept) . . . . . .          .          .          .
## va3         . . . . . . -0.03520319 -0.07252628 -0.1034437 -0.1249483
## irf3        . . . . . .  .          .          .          0.5718534
## rpe3        . . . . . .  .          .          .         -0.7860372
## srf3        . . . . . .  .          .          .          0.1183151
## ped3        . . . . . .  .          .          .          .
## shrm3       . . . . . .  .          .          0.2331034  0.5079736
## irhr3       . . . . . .  .          .          .          .
##
## (Intercept)  .          .          .          .          .          .
## va3         -0.13975600 -0.1513371 -0.1598684 -0.1658216 -0.1697274 -0.1721965
## irf3         1.14494507  1.6117009  2.0392610  2.3518004  2.5656987  2.6968318
## rpe3        -3.06218192 -4.9470439 -6.2687697 -7.1440682 -7.6804136 -8.0319183
## srf3         0.28480080  0.3842431  0.5553757  0.6824097  0.7658543  0.8163302
## ped3         0.01397453  0.0677689  0.1038970  0.1295727  0.1452397  0.1538668
## shrm3        0.64584054  0.7005423  0.7362920  0.7470202  0.7628669  0.7771149
## irhr3        .           .          .          .          0.5738429  2.1988264
##
## (Intercept)  .          .          .
## va3         -0.1737509 -0.1747025 -0.1761429
## irf3         2.7372989  2.7203950  2.6917670
## rpe3        -8.2493884 -8.3827913 -8.5902831
## srf3         0.8466379  0.8663159  0.7251618
## ped3         0.1591355  0.1623673  0.1673054
## shrm3        0.7860722  0.7915390  0.8001859
## irhr3        3.2185034  3.8446911  4.7959249
##
## $`2`
## 8 x 22 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 22 column names 's0', 's1', 's2' ... ]]
##
## (Intercept) . . . . . .          .          .          .
## va3         . . . . . . -0.02057737 -0.04162276 -0.06025614 -0.07372702
## irf3        . . . . . .  .          .          .          .
## rpe3        . . . . . .  .          .          .          .
## srf3        . . . . . .  .          .          .          .
## ped3        . . . . . .  .          .          .          .
## shrm3       . . . . . .  .          .          .          .
## irhr3       . . . . . .  .          .          .          .
##
## (Intercept)  .          .          .          .          .
## va3         -0.08282329 -0.0900876 -0.09541099 -0.09908192 -0.1015582
## irf3         .          .          .          .          .
## rpe3        -0.87031453 -1.6052927 -2.15596754 -2.51931138 -2.7260805
## srf3         .          .          0.13011230  0.23155354  0.2989920
## ped3         .          .          .          .          .
## shrm3        .          .          .          .          .
## irhr3        .          .          0.61311924  1.05539464  1.5986137
##
## (Intercept)  .          .          .          .
## va3         -0.1031526 -0.1041624 -0.1047831 -0.1057184
## irf3         .         -0.0493983 -0.1328163 -0.2613658
```

```
## rpe3        -2.8636436 -2.9485599 -3.0008127 -3.0820727
## srf3         0.3374638  0.3605606  0.3758175  0.2284724
## ped3         .          .          .          .
## shrm3        .          .          .          .
## irhr3        2.5824482  3.2063948  3.5930518  4.1679783
##
## $`3`
## 8 x 22 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 22 column names 's0', 's1', 's2' ... ]]
##
## (Intercept) . . . . . . . . . . . . .          .          .
## va3         . . . . . . . . . . . . .          .          .
## irf3        . . . . . . . . . . . . .         -0.2811918 -1.09983208
## rpe3        . . . . . . . . . . . . .          .          .
## srf3        . . . . . . . . . . . . .          .         -0.01539244
## ped3        . . . . . . . . . . . . .          .         -0.01897004
## shrm3       . . . . . . . . . . . . .          .          .
## irhr3       . . . . . . . . . . . . -0.8726986 -5.6759699 -9.03318401
##
## (Intercept)  .              .            .             .             .
## va3          .              .            .             .             .
## irf3        -1.684391e+00  -2.11263627  -2.380802285  -2.546271966  -2.67277158
## rpe3         .              .            .             .             .
## srf3         .              .            .             .            -0.00140484
## ped3        -2.298263e-02  -0.01416308  -0.008842146  -0.009643775  -0.01006091
## shrm3       -4.642663e-04  -0.05583025  -0.073173302  -0.081289405  -0.08672011
## irhr3       -1.112575e+01 -12.53269304 -13.255194664 -13.352067305 -13.40822185
##
## (Intercept)   .             .
## va3           .             .
## irf3         -2.772479995  -2.924361123
## rpe3          .             .
## srf3         -0.004788294  -0.179539796
## ped3         -0.009908114  -0.009977214
## shrm3        -0.090455058  -0.096453136
## irhr3       -13.451128717 -13.571889387
##
## $`4`
## 8 x 22 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 22 column names 's0', 's1', 's2' ... ]]
##
## (Intercept) . . . . . . . .          .          .          .
## va3         . . . . . . . . 0.008114908 0.03931856 0.05194151 0.06153699
## irf3        . . . . . . . .          .          .          .
## rpe3        . . . . . . . .          .          .          0.52123626
## srf3        . . . . . . . .          .          .          .
## ped3        . . . . . . . .          .          .          .
## shrm3       . . . . . . . .          .          .          .
## irhr3       . . . . . . . .          .          .          .
##
## (Intercept) .           .           .           .           .
## va3         0.06958817  0.07893575  0.08993163  0.09915386  0.10618300
```

```
## irf3        .          .          .          .          .
## rpe3        1.78299102  2.63216017  2.86850949  2.98083841  3.06841127
## srf3        .          .          .          .          .
## ped3        .          .          .          .          0.03756685
## shrm3       .          -0.08393455 -0.28697122 -0.45364982 -0.59689138
## irhr3       .          .          .          .          .
##
## (Intercept)  .          .          .          .          .          .
## va3          0.11113368  0.1144478  0.1164983  0.1177648  0.1185682  0.1198228
## irf3         .          .          .          .          .          .
## rpe3         3.16311224  3.2989525  3.3978635  3.4602312  3.4982182  3.5555475
## srf3        -0.08454462 -0.1426642 -0.1771895 -0.2023310 -0.2211291 -0.4158130
## ped3         0.08349315  0.1116582  0.1211924  0.1274663  0.1318986  0.1382098
## shrm3       -0.71883789 -0.7687373 -0.7967989 -0.8159723 -0.8289857 -0.8519489
## irhr3        .          .          .          .          .          .
##
## $`5`
## 8 x 22 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 22 column names 's0', 's1', 's2' ... ]]
##
## (Intercept) . . . . . .          .          .          .
## va3         . . . . . . 0.03899083 0.1164836 0.1887788 0.2520964
## irf3        . . . . . .          .          .          .
## rpe3        . . . . . .          .          .          .
## srf3        . . . . . .          .          .          .
## ped3        . . . . . .          .          .          .
## shrm3       . . . . . .          .          .          .
## irhr3       . . . . . .          .          .          .
##
## (Intercept) .          .          .          .          .          .
## va3         0.3054276 0.3476772  0.3794623  0.4016800  0.4167198  0.4434078
## irf3        .          0.3127149  0.6374266  0.8001842  0.9112436  1.2100647
## rpe3        .          0.7123023  1.0279928  1.2187905  1.3440130  1.6101220
## srf3        .          .          .          .          .          .
## ped3        .          .          -0.1423959 -0.2421719 -0.3063819 -0.4172700
## shrm3       0.2002858 0.5629870  0.8378496  1.0060851  1.1107866  1.2803214
## irhr3       .          .          .          .          -0.8405746 -9.4047987
```

#Finding optimal lambda

```
set.seed(123)
cv.lso2 <- cv.glmnet(xxx,yyy,family="multinomial",type.multinomial = "grouped", alpha = 1, intercept = T
```

```
## Warning: from glmnet C++ code (error code -63); Convergence for 63th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet C++ code (error code -65); Convergence for 65th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

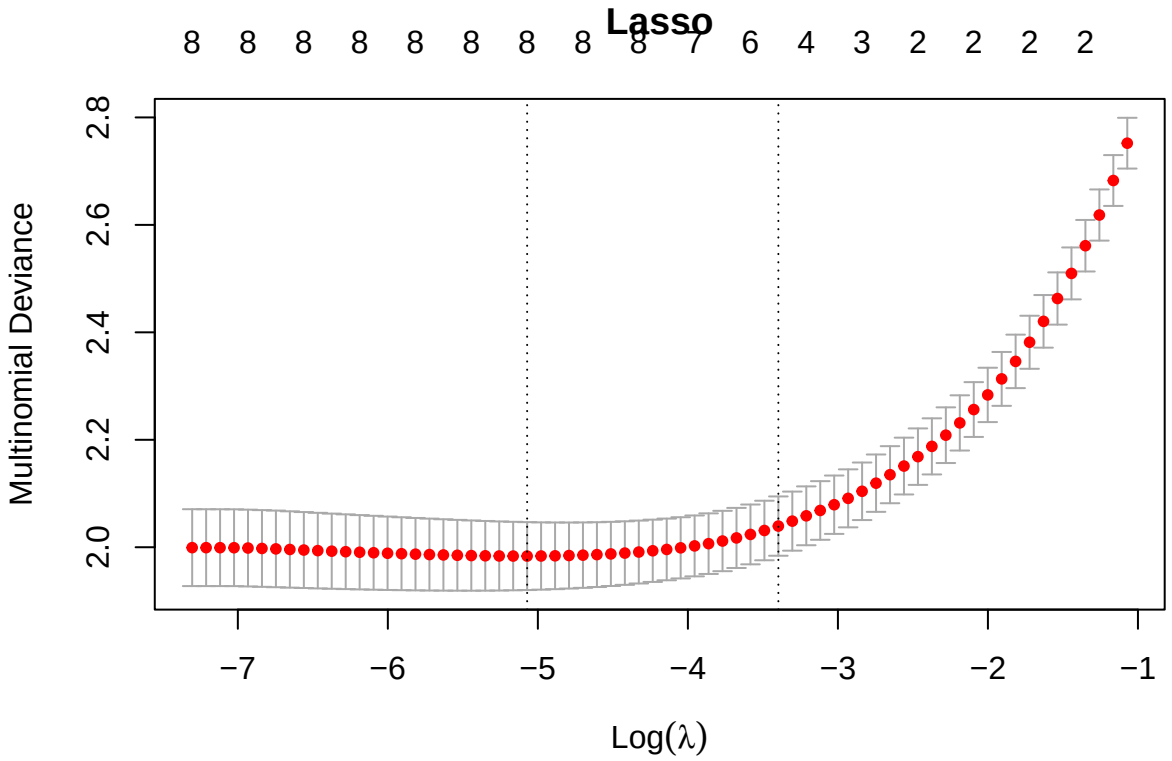
```
(lmin<- cv.lso2$lambda.min)
```

```
## [1] 0.006272975
```

```
log(cv.lso2$lambda.min)
```

```
## [1] -5.071505
```

```
index <- which(cv.lso2$lambda == lmin)
plot(cv.lso2, main="Lasso")
```




```
l <- cv.lso2$glmnet.fit$beta

l2 <- rbind(l$'1'[,index], l$'2'[,index], l$'3'[,index], l$'4'[,index], l$'5'[,index])
kable(cbind(1:5,l2), col.names = c("Class",lab), caption = "Training Set: LASSO Coefficients") %>%
  kable_styling(latex_options = c("striped")) %>%
  landscape
```

Table 6: Training Set: LASSO Coefficients

Class	Incercept	VA3	IRF3	RPE3	SRF3	PED3	SHRM3	IRHR3
1	0	-0.1813579	1.7106667	-5.344604	0.4358667	0.0676935	0.6514925	2.5592639
2	0	-0.1158059	-0.3613421	-1.517477	0.0597846	-0.0336802	-0.0730981	4.0446103
3	0	-0.0177036	-1.7950970	0.930346	-0.1843302	-0.0449679	-0.1380812	-7.7755220
4	0	0.0917282	-0.1712214	4.082223	-0.2318337	0.0378429	-0.6815918	1.8339265
5	0	0.2231393	0.6169937	1.849513	-0.0794874	-0.0268882	0.2412787	-0.6622787

Table 7: Test Set: Predicted vs. True classification

	True Class				
Predicted Class	1	2	3	4	5
1	3	2	0	0	0
2	10	38	10	10	0
3	2	9	17	11	0
4	0	4	16	49	5

Table 8: Training Set: Predicted vs. True classification

	True Class				
Predicted Class	1	2	3	4	5
1	19	9	0	0	0
2	45	160	57	15	0
3	2	30	61	31	0
4	2	23	74	198	14

Lasso Prediction

```
lasso3 <- glmnet(xxx,yyy, family = "multinomial", alpha = 1)
lso.train.pred <- predict(lasso3, newx =  model.matrix(~.,train[,xnams]), s=lmin, type = "class")
lso.test.pred <- predict(lasso3, newx =  model.matrix(~.,test[,xnams]), s=lmin, type = "class")

t1 <- table(Predicted = lso.train.pred,True = train$class)
t2 <- table(Predicted = lso.test.pred, True = test$class)

kable(cbind(1:4,t2), caption = "Test Set: Predicted vs. True classification", col.names = c("Predicted C
  add_header_above(header = c(" ","True Class"=5))
```

```
kable(cbind(1:4,t1), caption = "Training Set: Predicted vs. True classification", col.names = c("Predic
  add_header_above(header = c(" ","True Class"=5))
```

```
##accuracy
sum(diag(t1))/sum(t1)
```

```
## [1] 0.5918919
```

```
sum(diag(t2))/sum(t2)
```

```
## [1] 0.5752688
```