Models template

Nick Borg

September 23, 2016

Contents

This is a generic template for predicting behavioral repsonses from fMRI data. We assume that you import behavioral and brain data from the same .csv file, in the format

`TR, Trial, Behavioral_1, ...., ROI_TR_1, ROI_TR_2, ... ROI2_TR_1, ...`

Library imports

```
library(dplyr)
library(lme4)
library(e1071)
library(lmtest)
library(lubridate)
library(knitr)
library(BaylorEdPsych)
```

Load your data.

```
data_path <- '/Users/kelly/cueexp/data/relapse_data/relapse_data_180330.csv'
setwd('/Users/kelly/cueexp/scripts/nick_relapse_scripts')
df <- read.csv(data_path)

#remove subjects without relapse data
df = df[is.na(df$relapse) == FALSE,]
#df <- filter(df, Subject !='tf151127', Subject!= 'er171009')

df$relapse <- as.factor(df$relapse)
df$relIn6Mos <- as.factor(df$relIn6Mos)
df$Subject <- df$subj
```

Clean your data

I recommend this extra preprocessing step if you end up using trial based data: You need to look at which columns contain brain data, here they are columns 19:113. the colnames function is useful for that. (Remember that R is 1 indexed.)

```
filterMotion <- function(mask, tr, condition){
  var <- paste0(mask,'_', condition, '_TR', tr)
  print(var)
  col <- df[var]
  print(sum(col>4))
  df[var][col>4] <- NA
  print(sum(is.na(df[var])))
}

#print(colnames(df))

for (mask in c('nacc','acc', 'ains', 'mpfc', 'vta')){
  for (tr in c("3", "4", "5", "6", '7', '567mean')){
    for( condition in c('drugs', 'food', 'neutral')){
      filterMotion(mask, tr, condition)
    }
  }
}
```

```
## [1] "nacc_drugs_TR3"
## [1] 0
## [1] 0
## [1] "nacc_food_TR3"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR3"
## [1] 0
## [1] 0
## [1] "nacc_drugs_TR4"
## [1] 0
## [1] 0
## [1] "nacc_food_TR4"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR4"
## [1] 0
## [1] 0
## [1] "nacc_drugs_TR5"
## [1] 0
## [1] 0
## [1] "nacc_food_TR5"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR5"
## [1] 0
## [1] 0
## [1] "nacc_drugs_TR6"
## [1] 0
```

```
## [1] 0
## [1] "nacc_food_TR6"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR6"
## [1] 0
## [1] 0
## [1] "nacc_drugs_TR7"
## [1] 0
## [1] 0
## [1] "nacc_food_TR7"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR7"
## [1] 0
## [1] 0
## [1] "nacc_drugs_TR567mean"
## [1] 0
## [1] 0
## [1] "nacc_food_TR567mean"
## [1] 0
## [1] 0
## [1] "nacc_neutral_TR567mean"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR3"
## [1] 0
## [1] 0
## [1] "acc_food_TR3"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR3"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR4"
## [1] 0
## [1] 0
## [1] "acc_food_TR4"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR4"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR5"
## [1] 0
## [1] 0
## [1] "acc_food_TR5"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR5"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR6"
## [1] 0
```

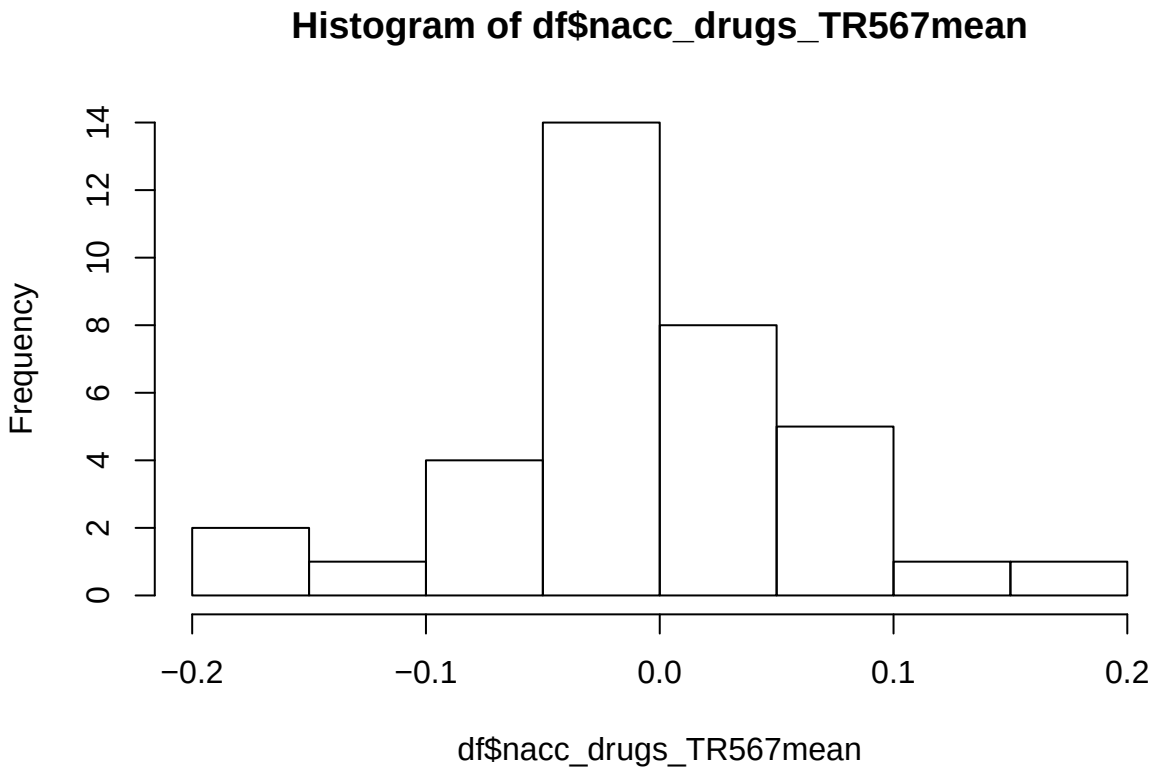
```
## [1] 0
## [1] "acc_food_TR6"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR6"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR7"
## [1] 0
## [1] 0
## [1] "acc_food_TR7"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR7"
## [1] 0
## [1] 0
## [1] "acc_drugs_TR567mean"
## [1] 0
## [1] 0
## [1] "acc_food_TR567mean"
## [1] 0
## [1] 0
## [1] "acc_neutral_TR567mean"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR3"
## [1] 0
## [1] 0
## [1] "ains_food_TR3"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR3"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR4"
## [1] 0
## [1] 0
## [1] "ains_food_TR4"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR4"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR5"
## [1] 0
## [1] 0
## [1] "ains_food_TR5"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR5"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR6"
## [1] 0
```

```
## [1] 0
## [1] "ains_food_TR6"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR6"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR7"
## [1] 0
## [1] 0
## [1] "ains_food_TR7"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR7"
## [1] 0
## [1] 0
## [1] "ains_drugs_TR567mean"
## [1] 0
## [1] 0
## [1] "ains_food_TR567mean"
## [1] 0
## [1] 0
## [1] "ains_neutral_TR567mean"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR3"
## [1] 0
## [1] 0
## [1] "mpfc_food_TR3"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR3"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR4"
## [1] 0
## [1] 0
## [1] "mpfc_food_TR4"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR4"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR5"
## [1] 0
## [1] 0
## [1] "mpfc_food_TR5"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR5"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR6"
## [1] 0
```

```
## [1] 0
## [1] "mpfc_food_TR6"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR6"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR7"
## [1] 0
## [1] 0
## [1] "mpfc_food_TR7"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR7"
## [1] 0
## [1] 0
## [1] "mpfc_drugs_TR567mean"
## [1] 0
## [1] 0
## [1] "mpfc_food_TR567mean"
## [1] 0
## [1] 0
## [1] "mpfc_neutral_TR567mean"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR3"
## [1] 0
## [1] 0
## [1] "vta_food_TR3"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR3"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR4"
## [1] 0
## [1] 0
## [1] "vta_food_TR4"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR4"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR5"
## [1] 0
## [1] 0
## [1] "vta_food_TR5"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR5"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR6"
## [1] 0
```

```
## [1] 0
## [1] "vta_food_TR6"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR6"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR7"
## [1] 0
## [1] 0
## [1] "vta_food_TR7"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR7"
## [1] 0
## [1] 0
## [1] "vta_drugs_TR567mean"
## [1] 0
## [1] 0
## [1] "vta_food_TR567mean"
## [1] 0
## [1] 0
## [1] "vta_neutral_TR567mean"
## [1] 0
## [1] 0
```

```
hist(df$nacc_drugs_TR567mean)
```




```
df$Subject <- df$subjid
```

```
evalClassifier <- function(df, formula, type='glm', split.method='8020', sep.subjects=FALSE, print.model
  # first get rid of any rows in which one of our variables is NA)
  formula.vars <- unlist(strsplit(gsub("[^[:alnum:]|'_' ]", "", as.character(formula)), c(' ')))
  for(v in formula.vars){
    if(v %in% colnames(df)) {
      df <- df[!is.na(df[v]), ]
    }
  }
  # make our results data frame.
  d.res <<- data.frame()
  if(sep.subjects){
    print("No separate subject implementation here.")
  } else {
    if(split.method=='loso'){

            #eval_model on each subjects
      if(type=='glm'){
        print("Evaluating Entire Model")
        # first evaluate the entire model so as to print model summary information
        .evalClassifier(df, formula, type, split.method = 'none', print.model = print.model)
      }
      print("Beginning LOSO...")
      for(subj in unique(df$Subject)){
        print(nrow((filter(df, Subject==subj))))
        accuracies <<- .evalClassifier(df, formula, type, split.method, test.sub=subj, print.model = pri
        d.res<<- rbind(d.res, data.frame(Subject=subj, accuracies))
        break
      }
      d.res <- d.res[order(-d.res$test.accuracy),] # order the subjects by their classifier accuracy
      # add the average of all the subjects
      d.res<- rbind(d.res, data.frame(Subject='Average', t(colMeans(d.res[,-1]))))
    } else{
      accuracies <- .evalClassifier(df, formula, type, split.method, print.model=TRUE)
      d.res <- rbind(d.res, data.frame(data='All', accuracies))
    }
  }
  d.res
}
```

.evalClassifier is defined like so:

```
.evalClassifier <- function(df, formula, type='glm', split.method='loso', test.sub = '',  print.model=F
  y <- all.vars(formula)[1] # get the y variable name to do train test splits

  if(split.method=='none' && type=='glm'){
    m <- fitModel(df, df, formula, type, print.model=TRUE)
  } else {
  train.test <<- trainTestSplit(df, split.method, y=y,test.sub=test.sub)
  m <- fitModel(df, train.test$train, formula, type, print.model)
  train.accuracy <- modelAccuracy(m, y, train.test$train, type)

  if(split.method=='loso'){
    test.accuracy  <- modelAccuracy(m, y, train.test$test, type, one_prediction=T)
    return( list('train.accuracy'=train.accuracy, 'test.accuracy'=test.accuracy[1], 'prediction'=test.a
```

```
  } else{
    test.accuracy  <- modelAccuracy(m, y, train.test$test, type)
  }
   list('train.accuracy'=train.accuracy, 'test.accuracy'=test.accuracy[1])
  }
}
```

Train test splits...

How your train test splits are defined will depend on your project, so you'll probably need to add cases to the following function.Right now there's just a basic function to call for

```
trainTestSplit <- function(df, split.method='8020', y='result', test.sub="", downsample=TRUE){
  set.seed(0)
  # Assume that the two responses are levels 1, 2 of a factor
  ind1 <- which(as.numeric(df[,y])==2)
  ind0 <- which(as.numeric(df[,y])==1)

  # downsample results. to upsample, use max and replace = TRUE
  # uncomment section below for one_subj
  if(downsample){
    sampsize <- min(length(ind1), length(ind0))
    sampind1 <- sample(ind1, sampsize, replace = FALSE)
    sampind0 <- sample(ind0, sampsize, replace = FALSE)
    sampind <- sample(c(sampind1,sampind0)) # the outside call to sample should randomize
  } else{
    sampind <- sample(1:nrow(df))
  }

  balanced.df <- df[sampind,]

  #Change response levels to 0,1 from 1, 2
  balanced.df[,y] <- factor(as.numeric(balanced.df[,y])-1, levels=c(0,1))

  #return last day
  if(split.method=='8020'){
    nrows <- nrow(balanced.df)
    train.ind <- sample(1:nrows, size=floor(.8 * nrows))
    test.ind <- c(1:nrows)[!c(1:nrows) %in% train.ind]
    d.train = balanced.df[train.ind,]
    d.test  =balanced.df[test.ind,]
    l <- list("train" = d.train, "test" = d.test)
  } else if(split.method=='loso'){
    train.ind <- balanced.df$Subject!= test.sub
    test.ind <- balanced.df$Subject==test.sub
    d.train <<- balanced.df[train.ind,]
    d.test  <<- balanced.df[test.ind,]
    l <- list("train" = d.train, "test" = d.test)
  } else {
    print('trainTestSplit called with split.method != 8020 or loso')
  }
  l
```

```
}
```

Lastly the fit model and predict accuracy functions are below:

```
modelAccuracy <- function(m, y, d.test, type='glm', one_prediction=FALSE){

  if(type=='glm'){
    pred = predict(m, newdata = d.test, type = 'response', allow.new.levels = TRUE)
    pred.bin = factor(ifelse(pred >= 0.5,1,0), levels = c(0,1))
  }

  if(type=='svm'){
    pred = predict(m, d.test)
    pred.bin <- factor(as.numeric(pred), levels=c(1,2), labels=c(0,1))
  }

  if(type=='naivebayes'){
    pred = predict(m, d.test, type='raw')[,2]
    pred.bin = factor(ifelse(pred >= 0.5,1,0), levels = c(0,1))
  }

  if(one_prediction){
    prediction <- as.integer(pred.bin[1]) - 1
  }
  class.tab = table(real=d.test[,y], model=pred.bin)
  #print(class.tab)
  acc <- (class.tab[1,1])/sum(class.tab)
  try(acc <- acc +  class.tab[2,2]/sum(class.tab))

  if(one_prediction){
    return(c(acc, prediction, as.integer(d.test[,y][1])-1))
  }
  acc
}

fitModel <- function(df, d.train, formula, type='glm', print.model=TRUE){
  # glimpse(d.train)
  needs.multilevel <- '|' %in% unlist(strsplit(gsub("[^[:alnum:]|'_' ]", "", as.character(formula)), c(
  if(type=='glm'){
    if(needs.multilevel){
      null_formula <- as.formula(paste(as.character(formula)[2], '~', '(1 | Subject)'))
      model <- glmer(formula, data = df, family = binomial(link = logit), control=glmerControl(optimizer
      m2 <- glmer(null_formula, data = df, family = binomial(link = logit), control=glmerControl(optimi

      if(print.model){
        print(summary(model), correlation=T)
        if(needs.multilevel){
          print(rsquared.glmm(model))
        }
        print(lrtest(m2, model))
        print(anova(model))
        #print(confint(model))
      }
      model <- glmer(formula, data = d.train, family = binomial(link = logit), control=glmerControl(opti
```

```
    } else{
      null_formula <- as.formula(paste(as.character(formula)[2], '~', '1'))
      model <- glm(formula, data=df, family="binomial")
      m2 <- glm(null_formula, data=df, family="binomial")

      if(print.model){
        print(summary(model))
        print(PseudoR2(model))
        print(lrtest(m2, model))
        print(anova(model))
      }
      #print(summary(d.train))
      model <- glm(formula, data=d.train, family="binomial")
    }


  }
  if(type=='svm'){
     model <- svm(formula, data=d.train)
  }
  if(type=='naivebayes'){
    if(needs.multilevel){
      print('naiveBayes cannot handle interaction terms')
    } else{
      model <-  naiveBayes(formula, data=d.train)
    }
  }
  model
}
```

```
formula <- as.formula("relIn6Mos~ nacc_drugs_TR567mean + years_of_use + poly_drug_dep + craving + bdi +
res <- evalClassifier(df, formula, type='glm', split.method='loso')
```

```
## [1] "Evaluating Entire Model"
##
## Call:
## glm(formula = formula, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0155  -0.6360  -0.1559   0.4902   1.8067
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.877e+00  4.268e+00   0.674   0.5003
## nacc_drugs_TR567mean  2.609e+01  1.370e+01   1.905   0.0568 .
## years_of_use         -6.342e-02  6.964e-02  -0.911   0.3625
## poly_drug_dep        -2.638e+00  2.080e+00  -1.269   0.2046
## craving              -2.367e-05  6.000e-01   0.000   1.0000
## bdi                   2.494e-02  5.996e-02   0.416   0.6775
## clinical_diag        -2.923e+00  1.889e+00  -1.548   0.1217
## age                   8.170e-02  8.454e-02   0.966   0.3338
## smoke                -3.195e+00  1.545e+00  -2.068   0.0386 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44.987  on 32  degrees of freedom
## Residual deviance: 25.516  on 24  degrees of freedom
## AIC: 43.516
##
## Number of Fisher Scoring iterations: 6
##
##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
##       0.43281504      -0.01175573       0.44569190       0.59891002
## McKelvey.Zavoina           Effron            Count        Adj.Count
##       0.68115134       0.46844607       0.81818182       0.57142857
##              AIC    Corrected.AIC
##      43.51607033      51.34215728
## Likelihood ratio test
##
## Model 1: relIn6Mos ~ 1
## Model 2: relIn6Mos ~ nacc_drugs_TR567mean + years_of_use + poly_drug_dep +
##     craving + bdi + clinical_diag + age + smoke
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   1 -22.494
## 2   9 -12.758  8 19.471    0.01253 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: relIn6Mos
##
## Terms added sequentially (first to last)
##
##
##                      Df Deviance Resid. Df Resid. Dev
## NULL                                    32     44.987
## nacc_drugs_TR567mean  1   1.8485        31     43.139
## years_of_use          1   1.9974        30     41.141
## poly_drug_dep         1   0.6374        29     40.504
## craving               1   0.0483        28     40.456
## bdi                   1   0.0117        27     40.444
## clinical_diag         1   5.4946        26     34.949
## age                   1   2.4791        25     32.470
## smoke                 1   6.9542        24     25.516
## [1] "Beginning LOSO..."
## [1] 1
```

```
kable(res)
```

Subject	train.accuracy	test.accuracy	prediction	ground.truth
ag151024	0.7407407	1	0	0
Average	0.7407407	1	0	0

```r
class.tab = table(model=res$prediction[1:nrow(res)-1],ground.truth= res$ground.truth[1:nrow(res)-1])
print(class.tab)
```

```
##      ground.truth
## model 0
##     0 1
```

```r
formula <- as.formula("relIn6Mos~ nacc_drugs_beta + years_of_use + bam_upset ")
res <- evalClassifier(df, formula, type='glm', split.method='loso')
```

```
## [1] "Evaluating Entire Model"
##
## Call:
## glm(formula = formula, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4993  -0.7735  -0.3717   0.7230   2.1368
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.38753    1.46142  -1.634    0.102
## nacc_drugs_beta 21.47490    8.34193   2.574    0.010 *
## years_of_use     0.06454    0.04281   1.508    0.132
## bam_upset        0.19429    0.35598   0.546    0.585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44.987  on 32  degrees of freedom
## Residual deviance: 30.290  on 29  degrees of freedom
## AIC: 38.29
##
## Number of Fisher Scoring iterations: 5
##
##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
##        0.3267008        0.1044154        0.3594157        0.4829741
## McKelvey.Zavoina           Effron            Count        Adj.Count
##        0.5165173        0.3844657        0.7878788        0.5000000
##              AIC    Corrected.AIC
##       38.2898542       39.7184257
## Likelihood ratio test
##
## Model 1: relIn6Mos ~ 1
## Model 2: relIn6Mos ~ nacc_drugs_beta + years_of_use + bam_upset
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   1 -22.494
## 2   4 -15.145  3 14.697   0.002094 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Deviance Table
##
## Model: binomial, link: logit
```

```
##
## Response: relIn6Mos
##
## Terms added sequentially (first to last)
##
##
##                 Df Deviance Resid. Df Resid. Dev
## NULL                                32     44.987
## nacc_drugs_beta  1  11.4915        31     33.496
## years_of_use     1   2.9086        30     30.587
## bam_upset        1   0.2972        29     30.290
## [1] "Beginning LOSO..."
## [1] 1
```

```
kable(res)
```

Subject	train.accuracy	test.accuracy	prediction	ground.truth
ag151024	0.8148148	1	0	0
Average	0.8148148	1	0	0

```
class.tab = table(model=res$prediction[1:nrow(res)-1],ground.truth= res$ground.truth[1:nrow(res)-1])
print(class.tab)
```

```
##      ground.truth
## model 0
##     0 1
```

{r} #formula <- as.formula("relIn6Mos~ nacc_drugs_TR567mean + age + smoke ") #res <- evalClassifier(df, formula, type='glm', split.method='loso') #kable(res) #class.tab = table(model=res$prediction res$ground.truth[1:nrow(res)-1]) #print(class.tab) #

```
formula <- as.formula("relIn6Mos~ nacc_drugs_beta + age ")
res <- evalClassifier(df, formula, type='glm', split.method='loso')
```

```
## [1] "Evaluating Entire Model"
##
## Call:
## glm(formula = formula, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4638  -0.7492  -0.2979   0.5707   2.0733
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.28781    1.73486  -1.895  0.05807 .
## nacc_drugs_beta 23.88361    8.86065   2.695  0.00703 **
## age              0.06431    0.03809   1.688  0.09136 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48.902  on 35  degrees of freedom
## Residual deviance: 31.601  on 33  degrees of freedom
## AIC: 37.601
##
## Number of Fisher Scoring iterations: 5
##
##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
##        0.3537883        0.1901955        0.3815762        0.5136135
## McKelvey.Zavoina           Effron            Count        Adj.Count
##        0.5835137        0.3875685        0.6944444        0.2666667
##              AIC    Corrected.AIC
##       37.6009890       38.3509890
## Likelihood ratio test
##
## Model 1: relIn6Mos ~ 1
## Model 2: relIn6Mos ~ nacc_drugs_beta + age
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   1 -24.451
## 2   3 -15.800  2 17.301   0.000175 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: relIn6Mos
##
## Terms added sequentially (first to last)
##
##
##                 Df Deviance Resid. Df Resid. Dev
## NULL                                35     48.902
## nacc_drugs_beta  1   14.055        34     34.847
## age              1    3.246        33     31.601
## [1] "Beginning LOSO..."
## [1] 1
```

```
kable(res)
```

Subject	train.accuracy	test.accuracy	prediction	ground.truth
ag151024	0.7586207	1	0	0
Average	0.7586207	1	0	0

```
class.tab = table(model=res$prediction[1:nrow(res)-1],ground.truth= res$ground.truth[1:nrow(res)-1])
print(class.tab)
```

```
##      ground.truth
## model 0
##     0 1
```