Multiclass logistic regression

Milica Cudina

For a similar analysis, look at this tutorial from UCLA. For more about `glmnet`, consult this tutorial by Trevor Hastie.

First, we import the data.

```r
data<-read.csv("hsbdemo.csv")
#View(data)
data
```
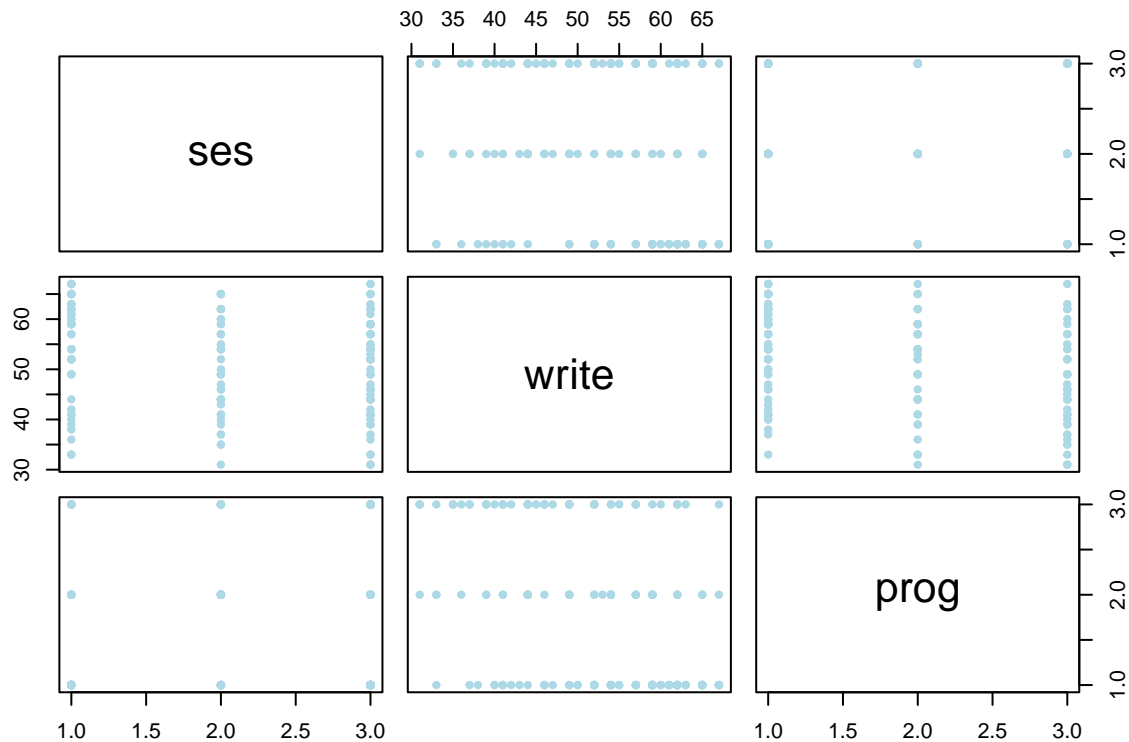
```
##   X  id female    ses schtyp    prog read write math science socst
## 1 1  45 female    low public vocation   34    35   41      29    26
## 2 2 108   male middle public  general   34    33   41      36    36
## 3 3  15   male   high public vocation   39    39   44      26    42
## 4 4  67   male    low public vocation   37    37   42      33    32
## 5 5 153   male middle public vocation   39    31   40      39    51
## 6 6  51 female   high public  general   42    36   42      31    39
## 7 7 164   male middle public vocation   31    36   46      39    46
##         honors awards cid
## 1 not enrolled      0   1
## 2 not enrolled      0   1
## 3 not enrolled      0   1
## 4 not enrolled      0   1
## 5 not enrolled      0   1
## 6 not enrolled      0   1
## 7 not enrolled      0   1
##  [ reached 'max' / getOption("max.print") -- omitted 193 rows ]
```

The data set contains variables on 200 students. We will focus on a small subset of predictors. The predictor variables will be socio-economic status `ses` (a three-level categorical variable) and writing score `write` (a quantitative variable). The outcome variable is program type `prog` (a three-level categorical variable). Since I am interested in just this subset, I will create a smaller data frame to analyze.

```r
df=data.frame(data$ses, data$write, data$prog)
colnames(df)<-c("ses", "write", "prog")
attach(df)
```

Some exploratory data analysis is called for. The first idea is, probably to try the `plot` command.

```r
plot(df,
     pch=20, col="lightblue")
```
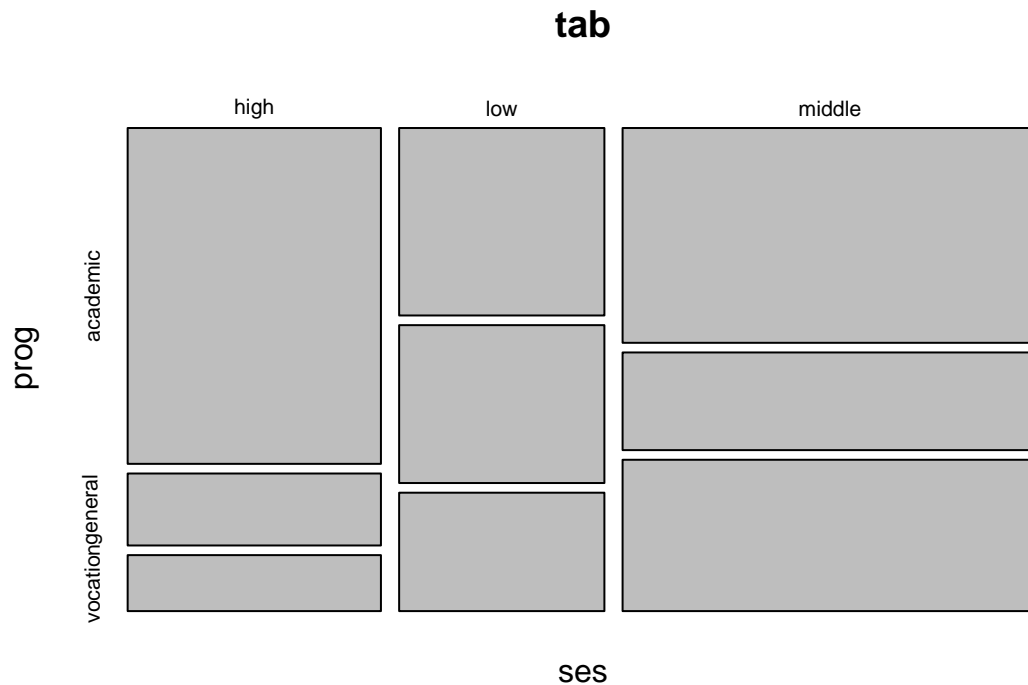
As we can see, this is not too useful. To look at the *association* between `ses` and `prog`, a two-way table might do the trick.

```
tab=table(ses, prog)
tab
```

```
##          prog
## ses       academic general vocation
##    high         42       9        7
##    low          19      16       12
##    middle       44      20       31
```

```
mosaicplot(tab)
```

tab

Which test might we use to test the independence hypothesis?

```
chi2<-chisq.test(tab)
chi2
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 16.604, df = 4, p-value = 0.002307
```
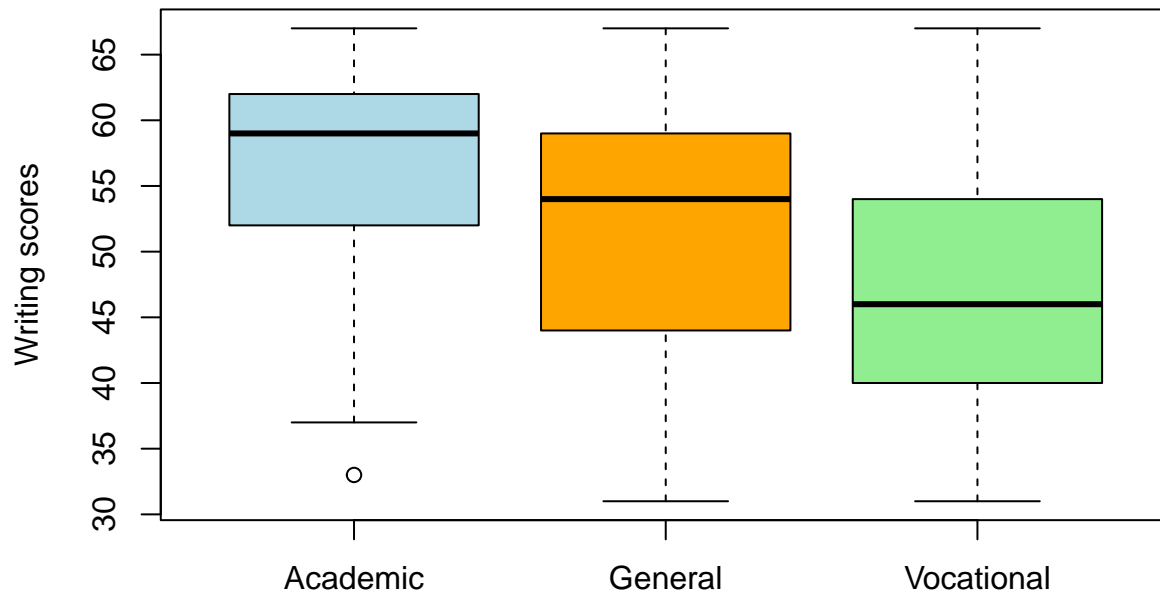
What about the association between the writing score and the program type?

Side-by-side boxplots might give insight.

```
write.ac=write[which(prog=="academic")]
write.gen=write[which(prog=="general")]
write.voc=write[which(prog=="vocation")]

boxplot(write.ac,write.gen, write.voc,
  main = "Writing scores by program type",
  ylab = "Writing scores",
  names = c("Academic", "General", "Vocational"),
  col=c("lightblue", "orange", "lightgreen")
)
```

Writing scores by program type

We see that there is an association, but we will learn more about the extent of the effect if we run a multiclass logistic regression.

multinom

Out first option is the `multinom` implementation in the `nnet` package.

```r
#install.packages("nnet")
library(nnet)
```

This implementation requires of us to set a *baseline* level for the response variable.

```r
df$prog<-as.factor(df$prog)
df$prog.base <- relevel(df$prog, ref = "academic")
```

Now, we can run the `multinom` function.

```r
mn.fit<-multinom(prog.base ~ ses + write, data=df)
```

```
## # weights:  15 (8 variable)
## initial  value 219.722458
## iter  10 value 179.983731
## final  value 179.981726
## converged
```

Here is what the object `mn.fit` looks like:

```r
summary(mn.fit)
```

```
## Call:
## multinom(formula = prog.base ~ ses + write, data = df)
##
## Coefficients:
##          (Intercept)    seslow sesmiddle       write
## general     1.689478 1.1628411 0.6295638 -0.05793086
## vocation    4.235574 0.9827182 1.2740985 -0.11360389
```

```
##
## Std. Errors:
##          (Intercept)    seslow sesmiddle       write
## general     1.226939 0.5142211 0.4650289 0.02141101
## vocation    1.204690 0.5955688 0.5111119 0.02222000
##
## Residual Deviance: 359.9635
## AIC: 375.9635
```

Importantly, the coefficients above are **different** from the coefficients one sees in our textbook. However, let us look at the fitted probabilities for our data set.

```
fitted.ps <- fitted(mn.fit)
fitted.ps
```

```
##       academic    general    vocation
## 1    0.1482721 0.33825094 0.51347695
## 2    0.1201988 0.18063349 0.69916776
## 3    0.4186768 0.23681368 0.34450947
## 4    0.1726839 0.35084330 0.47647284
## 5    0.1001206 0.16894278 0.73093666
## 6    0.3533583 0.23780466 0.40883700
## 7    0.1562526 0.19735515 0.64639223
## 8    0.1001206 0.16894278 0.73093666
## 9    0.2331247 0.22040133 0.54647393
## 10   0.1699365 0.20255764 0.62750590
## 11   0.2777676 0.37620933 0.34602312
## 12   0.2917517 0.23361220 0.47463615
## 13   0.1071652 0.30822610 0.58460873
## 14   0.2888732 0.22953872 0.48158807
## 15   0.1482721 0.33825094 0.51347695
## 16   0.2777676 0.37620933 0.34602312
## 17   0.3126201 0.37709158 0.31028831
## 18   0.3293850 0.23309576 0.43751925
## 19   0.3293850 0.23309576 0.43751925
## 20   0.6324633 0.20043456 0.16710218
## 21   0.1998541 0.21215684 0.58798906
## 22   0.2888732 0.22953872 0.48158807
## 23   0.3306561 0.37639804 0.29294591
## 24   0.2777676 0.37620933 0.34602312
## 25   0.1726839 0.35084330 0.47647284
## 26   0.3966353 0.23772687 0.36563784
## 27   0.3676891 0.37276358 0.25954735
## 28   0.2888732 0.22953872 0.48158807
## 29   0.2291956 0.36934430 0.40146013
## 30   0.3865745 0.36985112 0.24357436
## 31   0.2888732 0.22953872 0.48158807
## 32   0.1996861 0.36131931 0.43899458
## 33   0.2141360 0.36565697 0.42020707
##  [ reached 'max' / getOption("max.print") -- omitted 167 rows ]
```

Also importantly, we do not have $p-$values reported above. We can figure them out easily, though.

```
z.scores<- summary(mn.fit)$coefficients/summary(mn.fit)$standard.errors
p.val=2*(1-pnorm(abs(z.scores)))
p.val
```

```
##            (Intercept)      seslow sesmiddle           write
## general   0.1685163893 0.02373673 0.1757949 0.0068169138868
## vocation 0.0004382601 0.09893276 0.0126741 0.0000003176088
```

How well did we do on the training set?

```
pred.mn.tr=predict(mn.fit, newdata = df, "class")
#pred.mn.tr
ind.tr=(pred.mn.tr==prog)
mean(ind.tr)
```

```
## [1] 0.61
```

How would we predict for a test case? First, we need to define what the inputs for the test case would be.

```
new.data <- data.frame(ses = c("low", "middle", "high"), write = c(quantile(df$write, 0.25), median(df$w
```

Now, we can use the `predict` command to see what the predicted probabilities would be.

```
pred.mn=predict(mn.fit, newdata = new.data, "probs")
pred.mn
```

```
##       academic   general   vocation
## 25% 0.3081702 0.3771467 0.31468315
##       0.5049235 0.2247935 0.27028295
## 75% 0.8043171 0.1347768 0.06090615
```

glmnet

Here, we need to install and load a package create by Hastie et al.

```
#install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-10
```

We notice that we need to transform our categorical predictor using dummy variables. For that, the following library is useful:

```
library(caret)
```

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

```
## Loading required package: lattice
```

First, I will put all of my predictors into a separate data frame.

```
temp.x=data.frame(data$ses, data$write)
```

Now, I will use the command `dummyVars` to create dummy variables for my one categorical predictor `ses` in my array of predictors.

```
dummy.x <- dummyVars(" ~ .", data = temp.x)
x <- as.matrix(predict(dummy.x, newdata = temp.x))
print(x)
```

```
##       data.seshigh data.seslow data.sesmiddle data.write
## 1              0           1              0         35
## 2              0           0              1         33
## 3              1           0              0         39
```

```
## 4                    0                1                0            37
## 5                    0                0                1            31
## 6                    1                0                0            36
## 7                    0                0                1            36
## 8                    0                0                1            31
## 9                    0                0                1            41
## 10                   0                0                1            37
## 11                   0                1                0            44
## 12                   1                0                0            33
## 13                   0                1                0            31
## 14                   0                0                1            44
## 15                   0                1                0            35
## 16                   0                1                0            44
## 17                   0                1                0            46
## 18                   0                0                1            46
## 19                   0                0                1            46
## 20                   1                0                0            49
## 21                   0                0                1            39
## 22                   0                0                1            44
## 23                   0                1                0            47
## 24                   0                1                0            44
## 25                   0                1                0            37
##  [ reached 'max' / getOption("max.print") -- omitted 175 rows ]
```
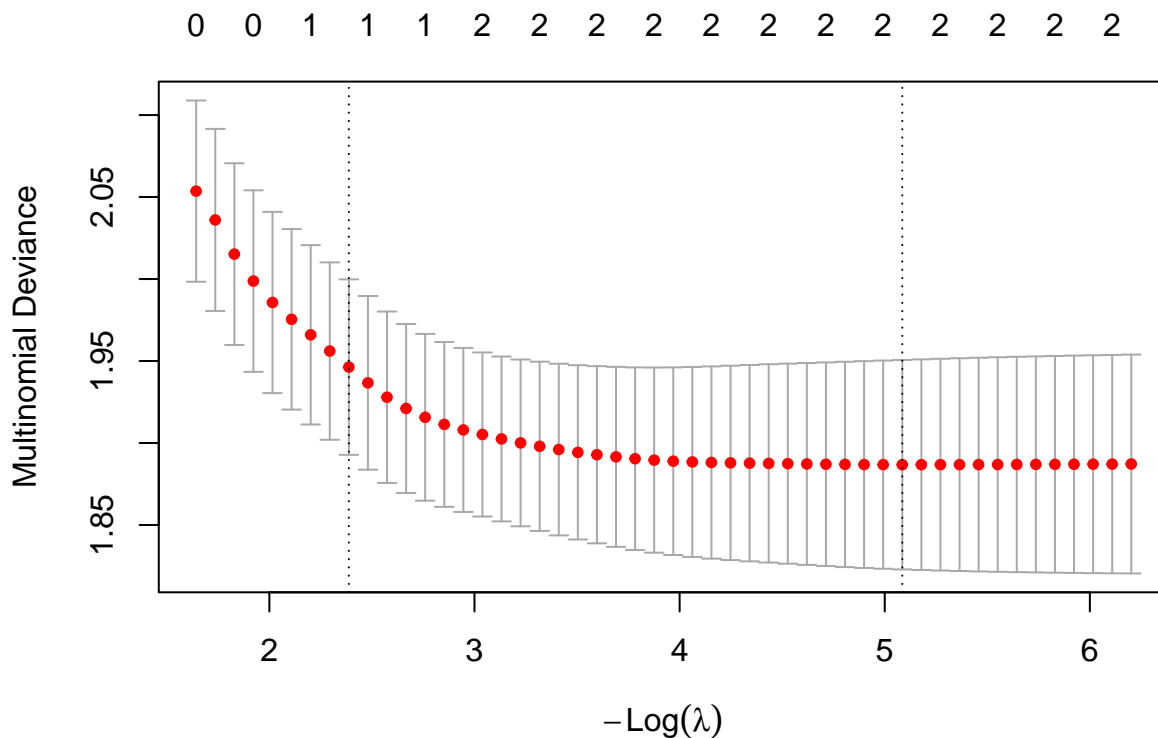
Now, we would like to fit the model using `glmnet`.

```r
y <- df$prog
#y
glm.fit <- glmnet(x, y, family = "multinomial")
#summary(glm.fit)
#glm.fit
predict(glm.fit, newx=x, s=0, type="response")
```

```
## , , s=0
##
##        academic     general    vocation
## 1     0.1526166 0.33690080 0.51048259
## 2     0.1240333 0.18489797 0.69106876
## 3     0.4172688 0.23461004 0.34812117
## 4     0.1771834 0.34881146 0.47400509
## 5     0.1037572 0.17343832 0.72280443
## 6     0.3528570 0.23557453 0.41156851
## 7     0.1602391 0.20117013 0.63859078
## 8     0.1037572 0.17343832 0.72280443
## 9     0.2368129 0.22328970 0.53989743
## 10    0.1739242 0.20620006 0.61987571
## 11    0.2822784 0.37220402 0.34551754
## 12    0.2921047 0.23156195 0.47633332
## 13    0.1110538 0.30824709 0.58069910
## 14    0.2919886 0.23186238 0.47614904
## 15    0.1526166 0.33690080 0.51048259
## 16    0.2822784 0.37220402 0.34551754
## 17    0.3169693 0.37272452 0.31030614
## 18    0.3319611 0.23508177 0.43295716
## 19    0.3319611 0.23508177 0.43295716
```

```
## 20   0.6288569 0.19944323 0.17169987
## 21   0.2037564 0.21543051 0.58081312
## 22   0.2919886 0.23186238 0.47614904
## 23   0.3348986 0.37189274 0.29320866
## 24   0.2822784 0.37220402 0.34551754
## 25   0.1771834 0.34881146 0.47400509
## 26   0.3955289 0.23549149 0.36897962
## 27   0.3716716 0.36807063 0.26025772
## 28   0.2919886 0.23186238 0.47614904
## 29   0.2338110 0.36607265 0.40011633
## 30   0.3904065 0.36510890 0.24448463
## 31   0.2919886 0.23186238 0.47614904
## 32   0.2042785 0.35863911 0.43708236
## 33   0.2187490 0.36267251 0.41857853
##
##   [ reached 'max' / getOption("max.print") -- omitted 167 rows ]
```

What if I try to find, by cross-validation, an optimal tuning parameter?

```
cv.fit <- cv.glmnet(x, y, family = "multinomial")
plot(cv.fit)
```


Now, we can try to predict at the optimal λ:

```
predict(cv.fit, newx = x, s = "lambda.min", type = "class")
```

```
##         lambda.min
##   [1,] "vocation"
##   [2,] "vocation"
##   [3,] "academic"
##   [4,] "vocation"
##   [5,] "vocation"
```

8

```
##    [6,]  "vocation"
##    [7,]  "vocation"
##    [8,]  "vocation"
##    [9,]  "vocation"
##   [10,]  "vocation"
##   [11,]  "general"
##   [12,]  "vocation"
##   [13,]  "vocation"
##   [14,]  "vocation"
##   [15,]  "vocation"
##   [16,]  "general"
##   [17,]  "general"
##   [18,]  "vocation"
##   [19,]  "vocation"
##   [20,]  "academic"
##   [21,]  "vocation"
##   [22,]  "vocation"
##   [23,]  "general"
##   [24,]  "general"
##   [25,]  "vocation"
##   [26,]  "academic"
##   [27,]  "academic"
##   [28,]  "vocation"
##   [29,]  "vocation"
##   [30,]  "academic"
##   [31,]  "vocation"
##   [32,]  "vocation"
##   [33,]  "vocation"
##   [34,]  "general"
##   [35,]  "vocation"
##   [36,]  "vocation"
##   [37,]  "vocation"
##   [38,]  "academic"
##   [39,]  "vocation"
##   [40,]  "vocation"
##   [41,]  "vocation"
##   [42,]  "academic"
##   [43,]  "academic"
##   [44,]  "academic"
##   [45,]  "general"
##   [46,]  "academic"
##   [47,]  "vocation"
##   [48,]  "academic"
##   [49,]  "vocation"
##   [50,]  "academic"
##   [51,]  "vocation"
##   [52,]  "vocation"
##   [53,]  "general"
##   [54,]  "academic"
##   [55,]  "academic"
##   [56,]  "academic"
##   [57,]  "academic"
##   [58,]  "academic"
##   [59,]  "academic"
```

```
##  [60,]  "vocation"
##  [61,]  "academic"
##  [62,]  "vocation"
##  [63,]  "academic"
##  [64,]  "academic"
##  [65,]  "academic"
##  [66,]  "vocation"
##  [67,]  "academic"
##  [68,]  "academic"
##  [69,]  "academic"
##  [70,]  "academic"
##  [71,]  "academic"
##  [72,]  "academic"
##  [73,]  "academic"
##  [74,]  "academic"
##  [75,]  "academic"
##  [76,]  "academic"
##  [77,]  "academic"
##  [78,]  "academic"
##  [79,]  "vocation"
##  [80,]  "vocation"
##  [81,]  "academic"
##  [82,]  "academic"
##  [83,]  "academic"
##  [84,]  "general"
##  [85,]  "academic"
##  [86,]  "general"
##  [87,]  "academic"
##  [88,]  "vocation"
##  [89,]  "academic"
##  [90,]  "vocation"
##  [91,]  "academic"
##  [92,]  "academic"
##  [93,]  "academic"
##  [94,]  "academic"
##  [95,]  "academic"
##  [96,]  "academic"
##  [97,]  "academic"
##  [98,]  "academic"
##  [99,]  "vocation"
## [100,]  "vocation"
##  [ reached 'max' / getOption("max.print") -- omitted 100 rows ]
```

We might want to see how well we did on the training data.

```
preds=predict(cv.fit, newx = x, s = "lambda.min", type = "class")
#preds
ind=(preds==y)
mean(ind)
```

```
## [1] 0.595
```