Predictions

2023-04-30

```
library(nnet)
library(ggplot2)
library(MASS)
library(knitr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-7
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(psych)
```

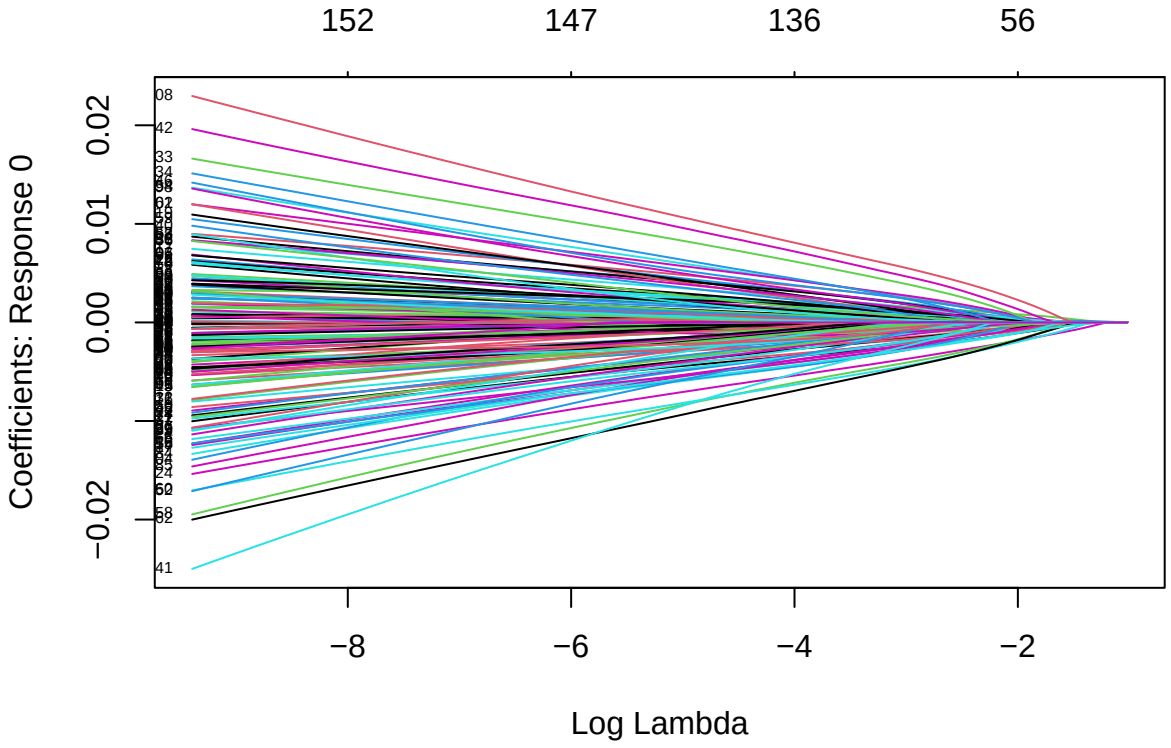
```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

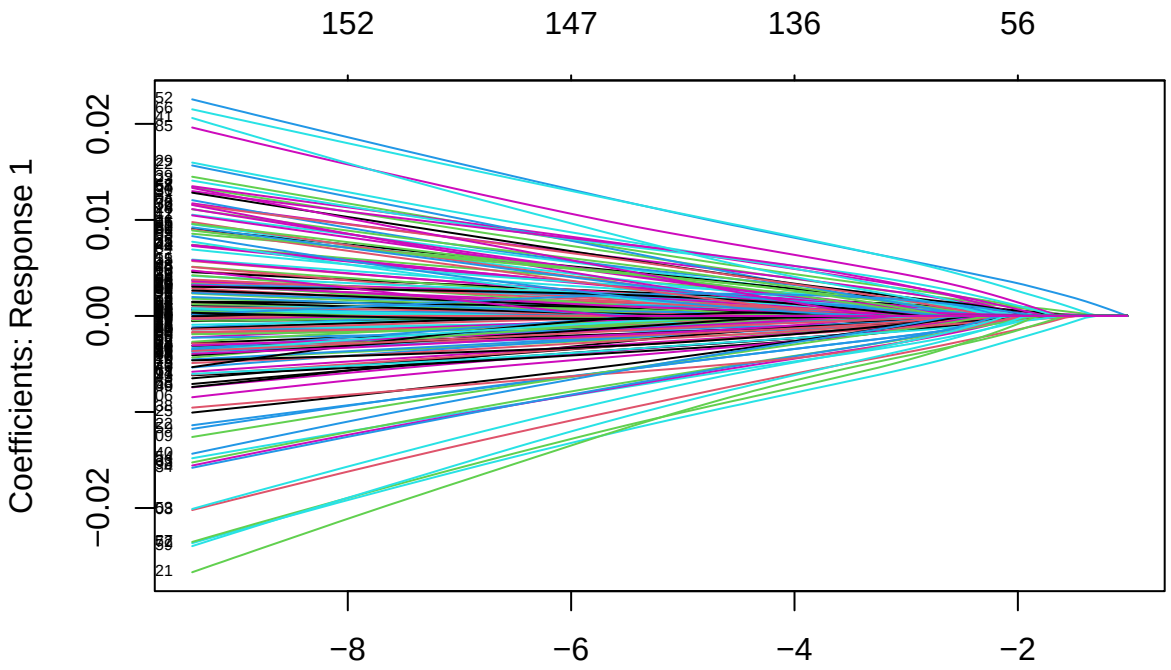
```
load(file = file.path(root_dir, "R_data", "pca_wm.Rda"))
load(file = file.path(root_dir, "R_data", "pca_gm.Rda"))
load(file = file.path(root_dir, "R_data", "pca_cb.Rda"))
```

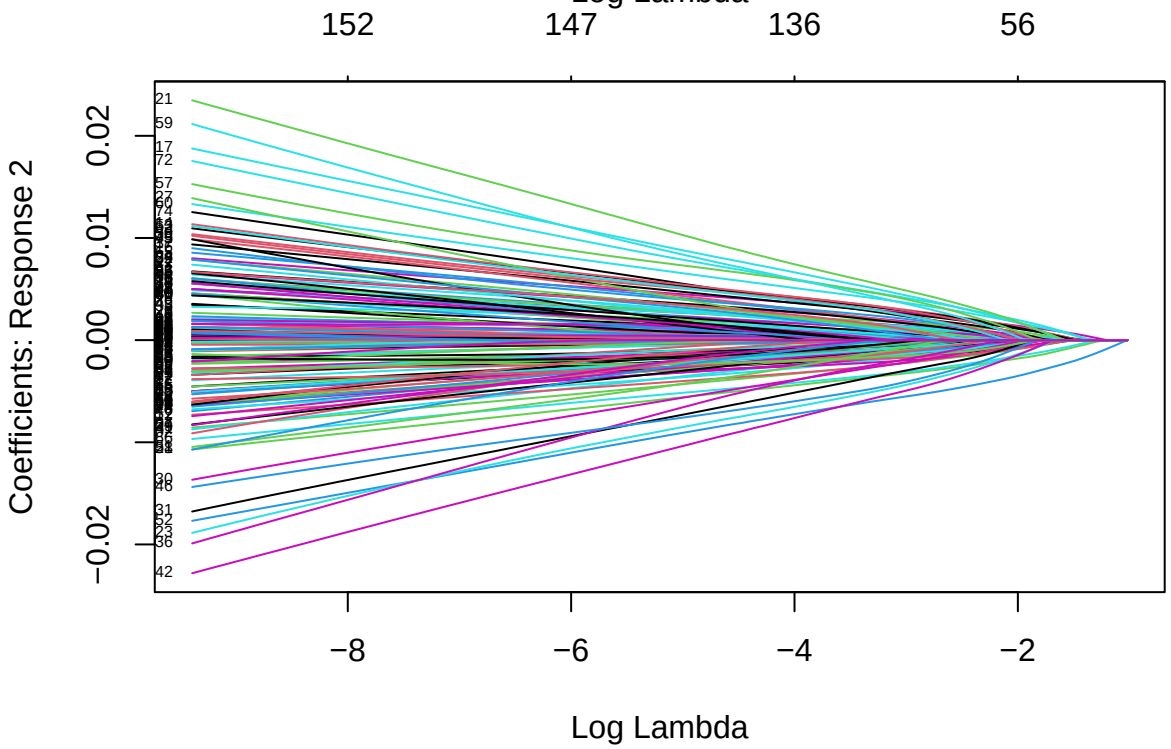
Running models

Because of our imbalanced dataset, in the multinomial logistic regression we adjust weights inversely proportional to class frequencies in the input data. The idea behind this weight adjustment is to ensure that each class contributes equally to the model fitting process, regardless of class imbalance in the input data. By assigning higher weights to minority classes and lower weights to majority classes, the model is able to give more importance to the minority classes, and avoid being biased towards the majority classes. We calculate this by n_samples / (n_classes * sum(I(y=class_j)))

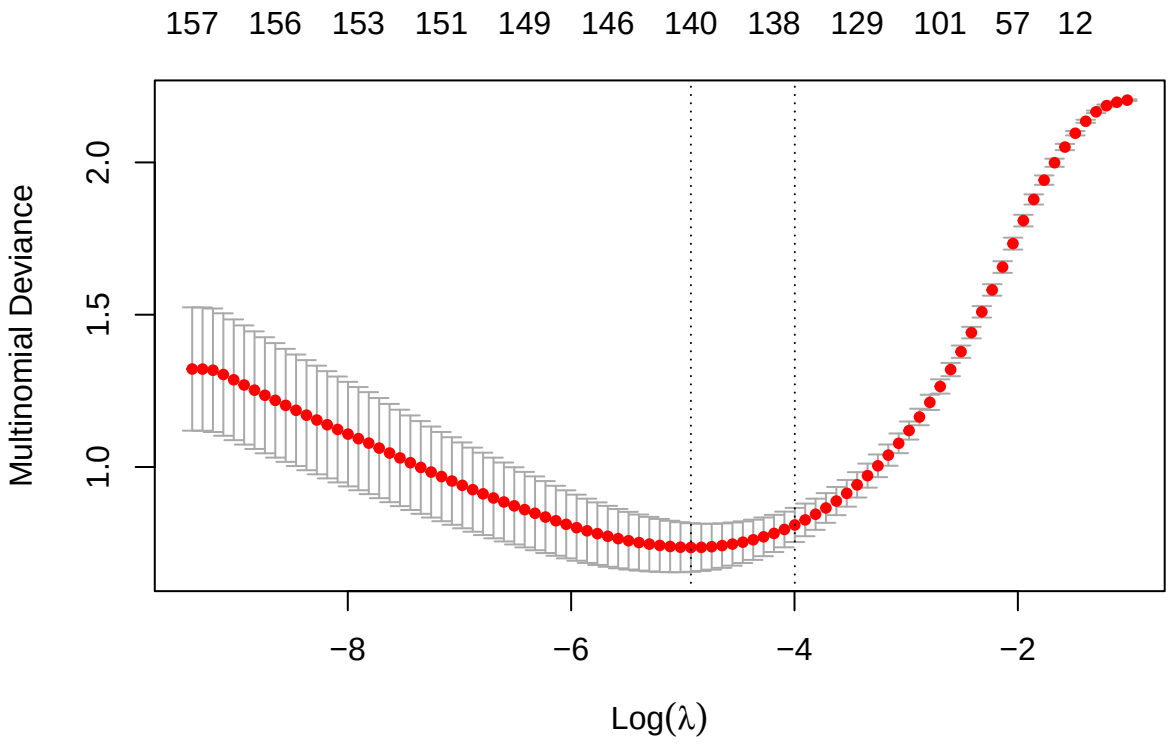
```
# balance weighted
wm_mn <- mn_reg(pca_wm)
```



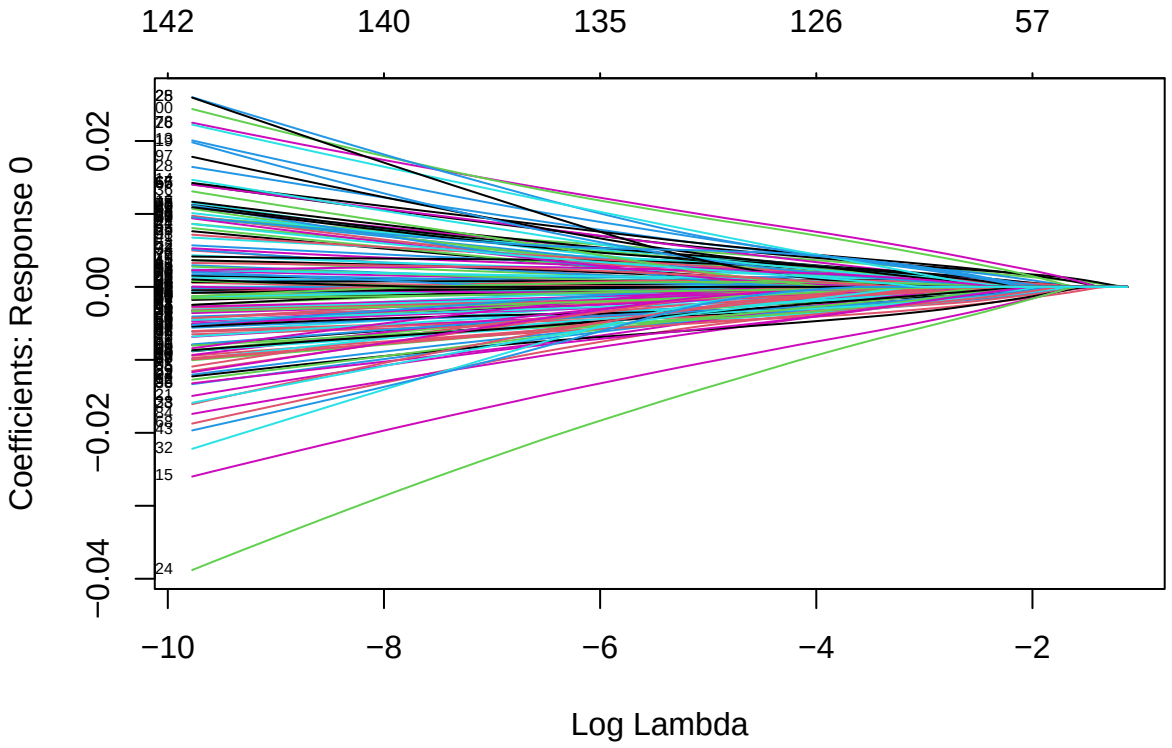


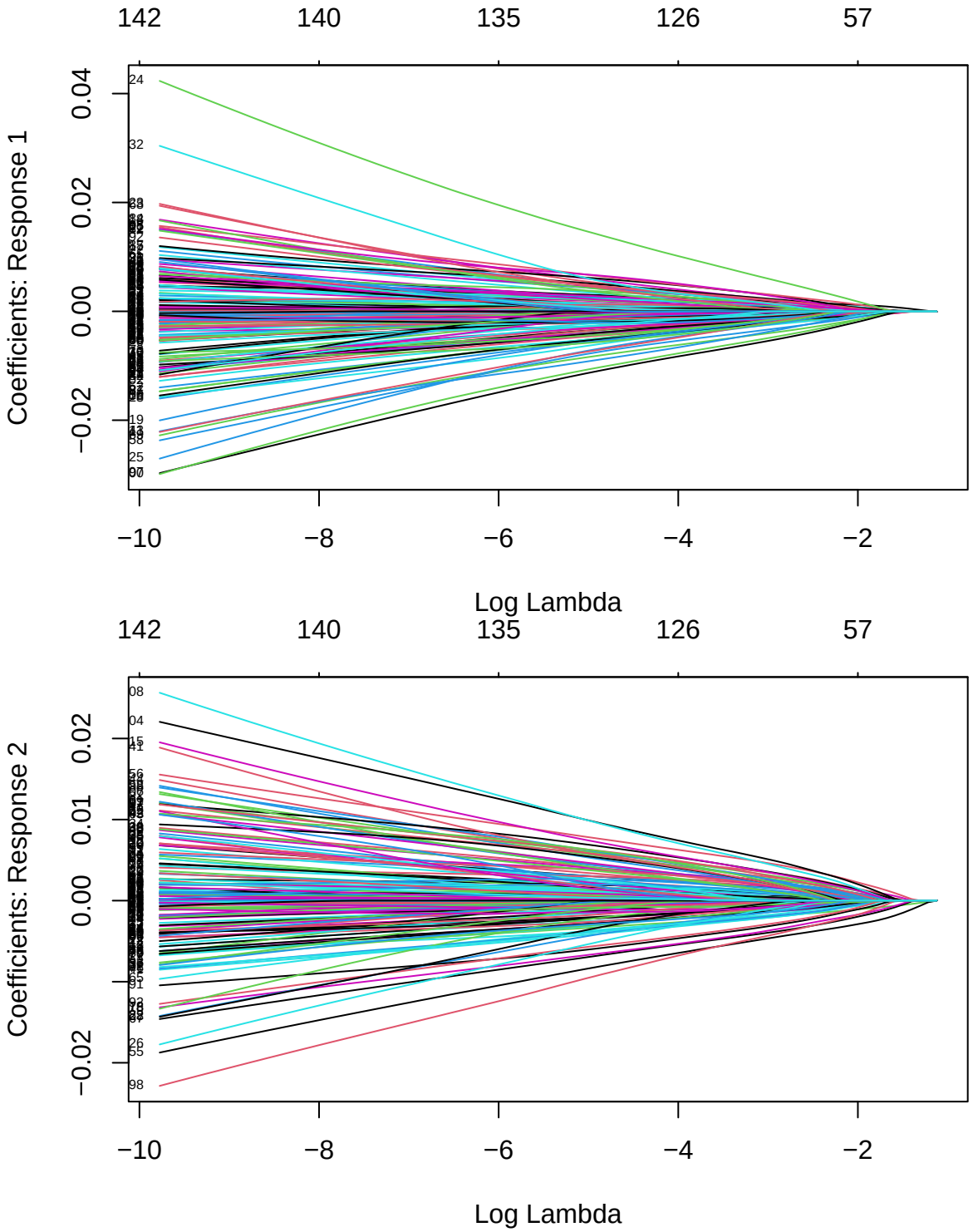



```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

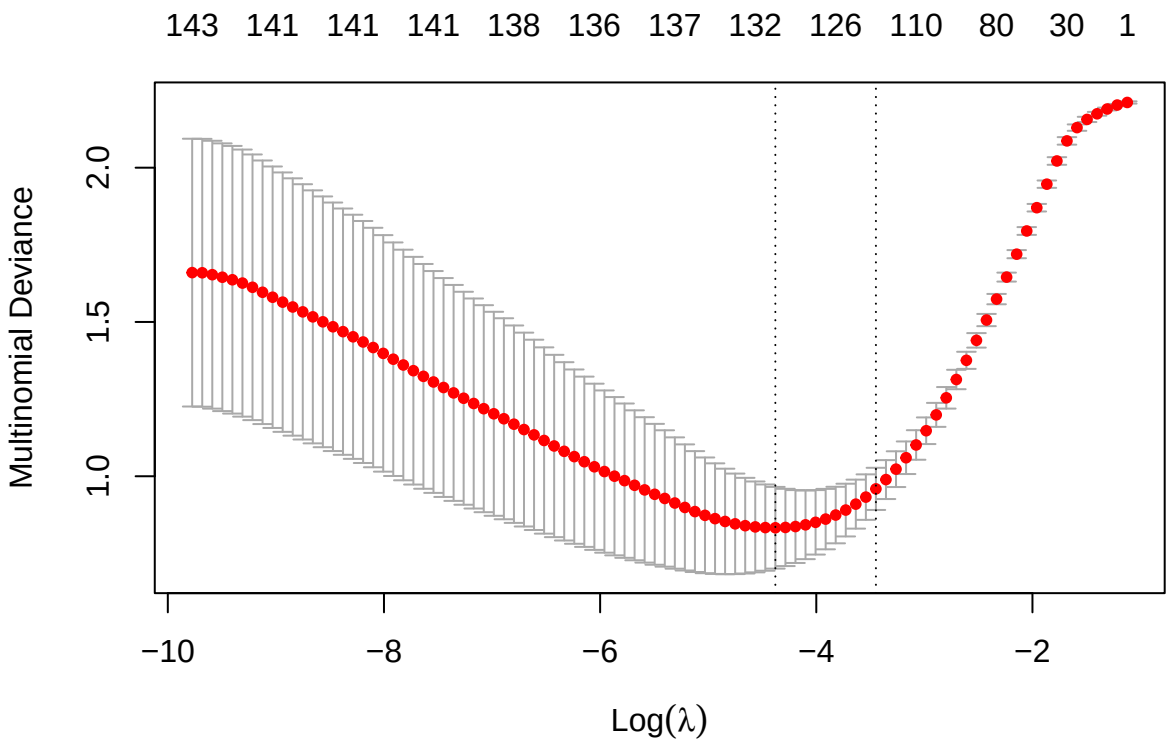



```
gm_mn <- mn_reg(pca_gm)
```

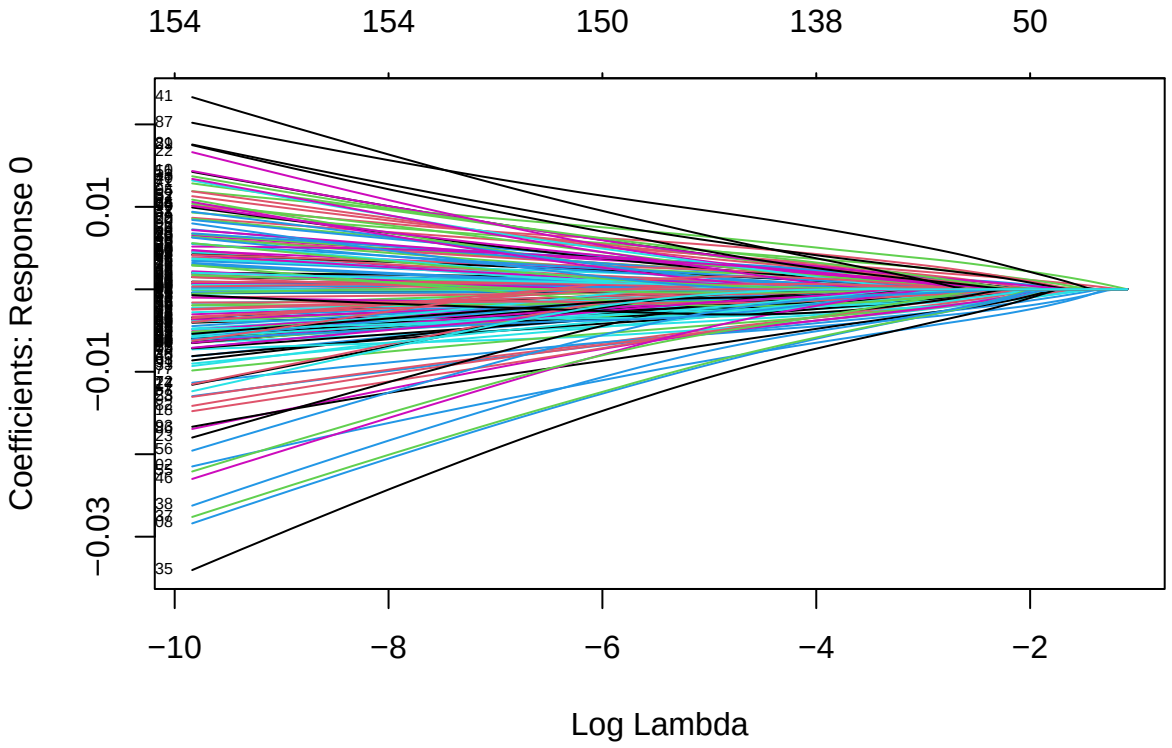


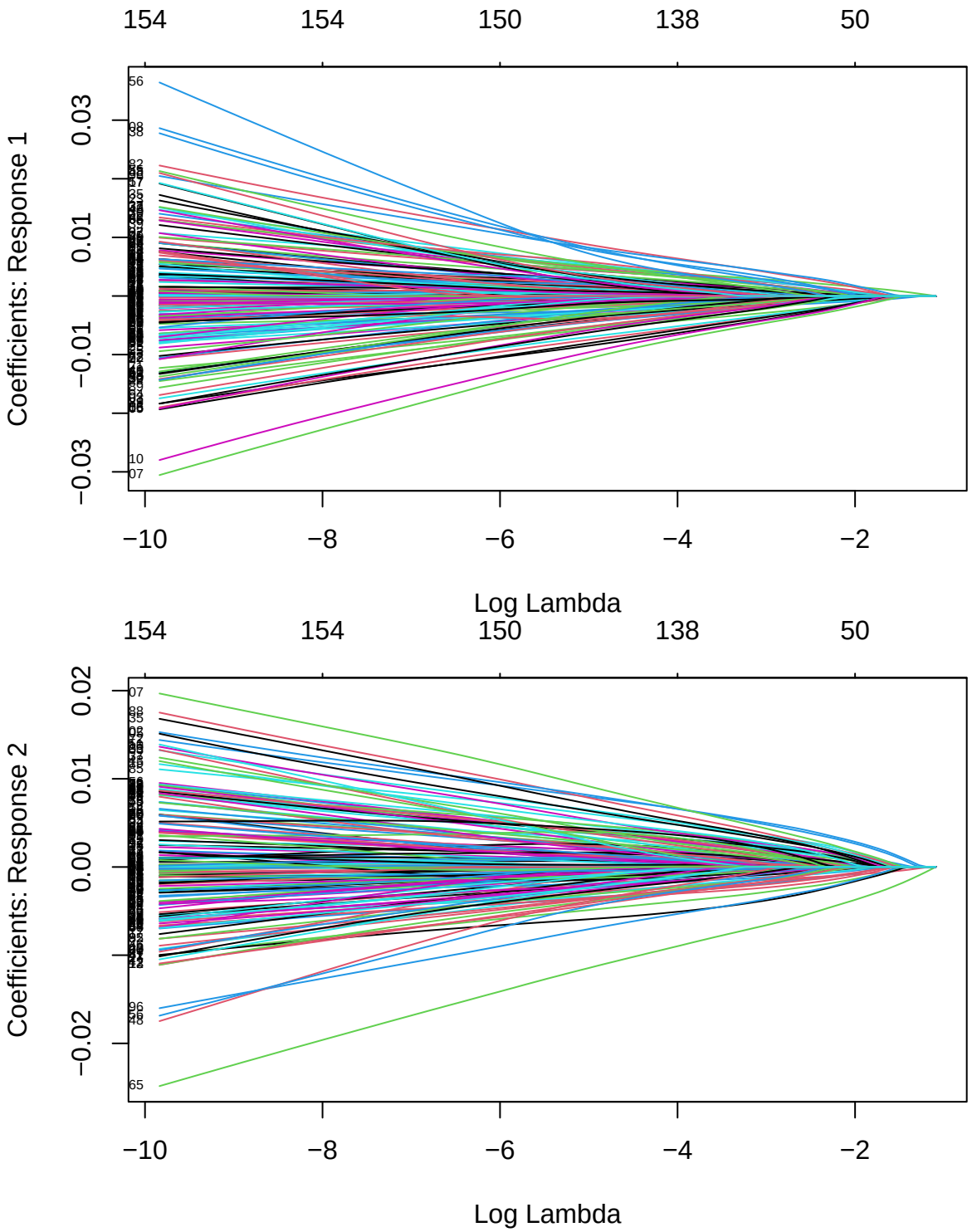



```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

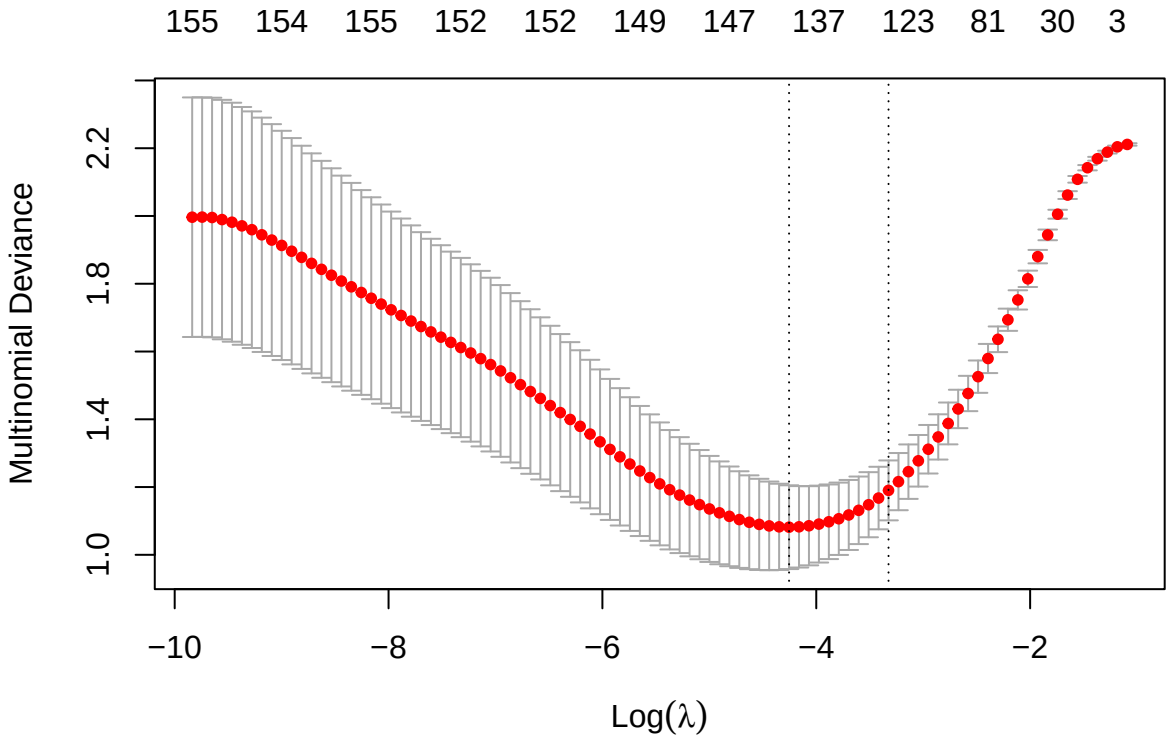



```
cb_mn <- mn_reg(pca_cb)
```

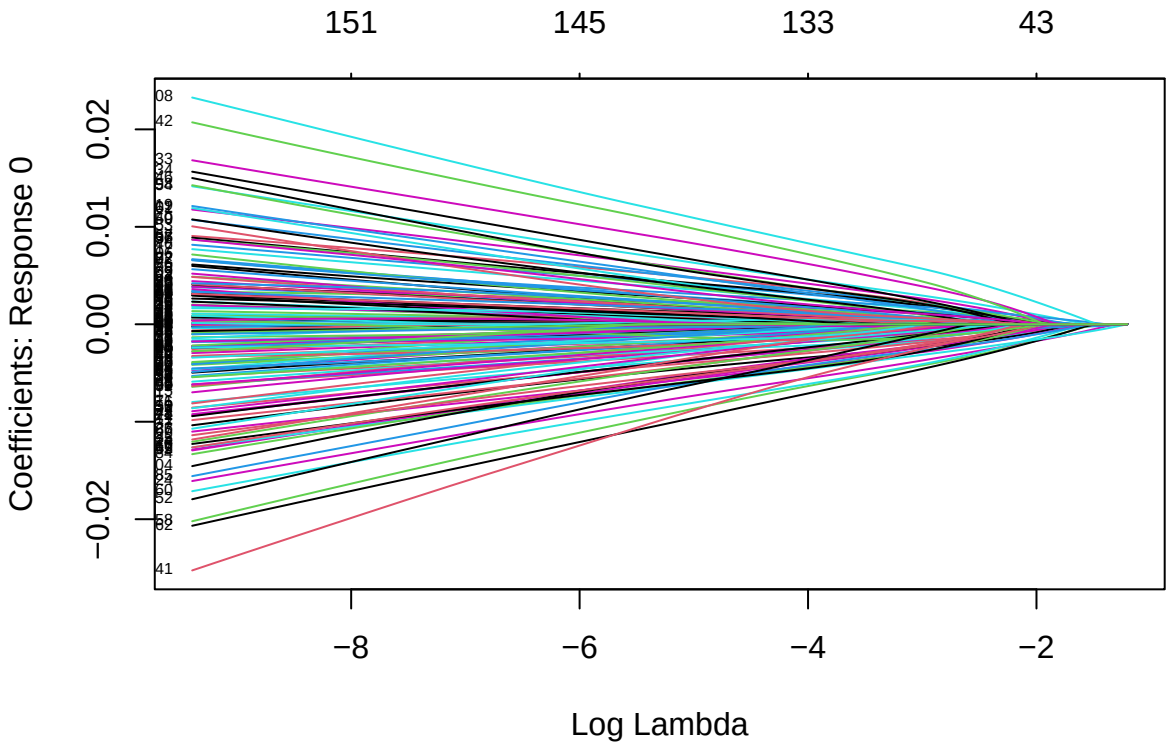


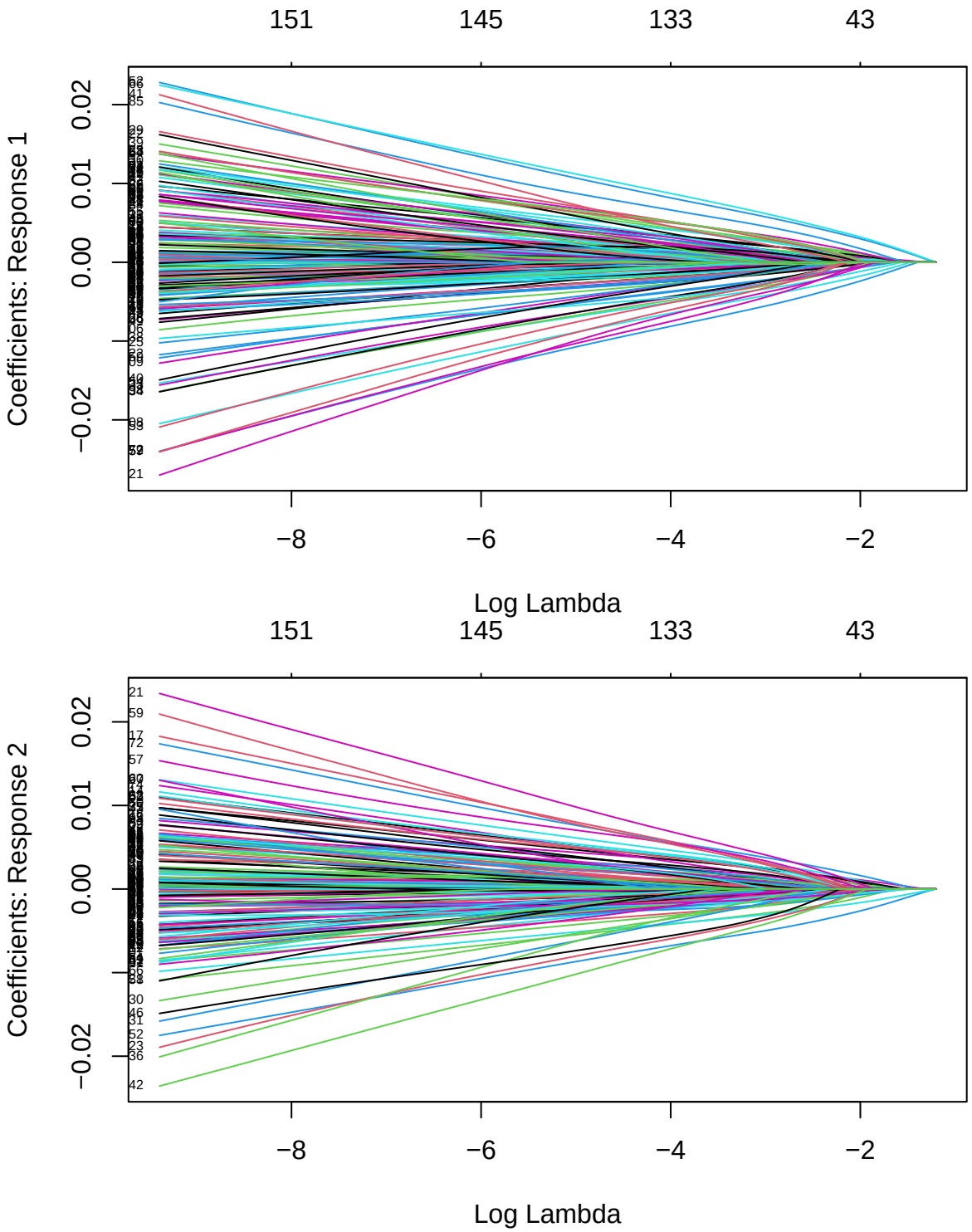



```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

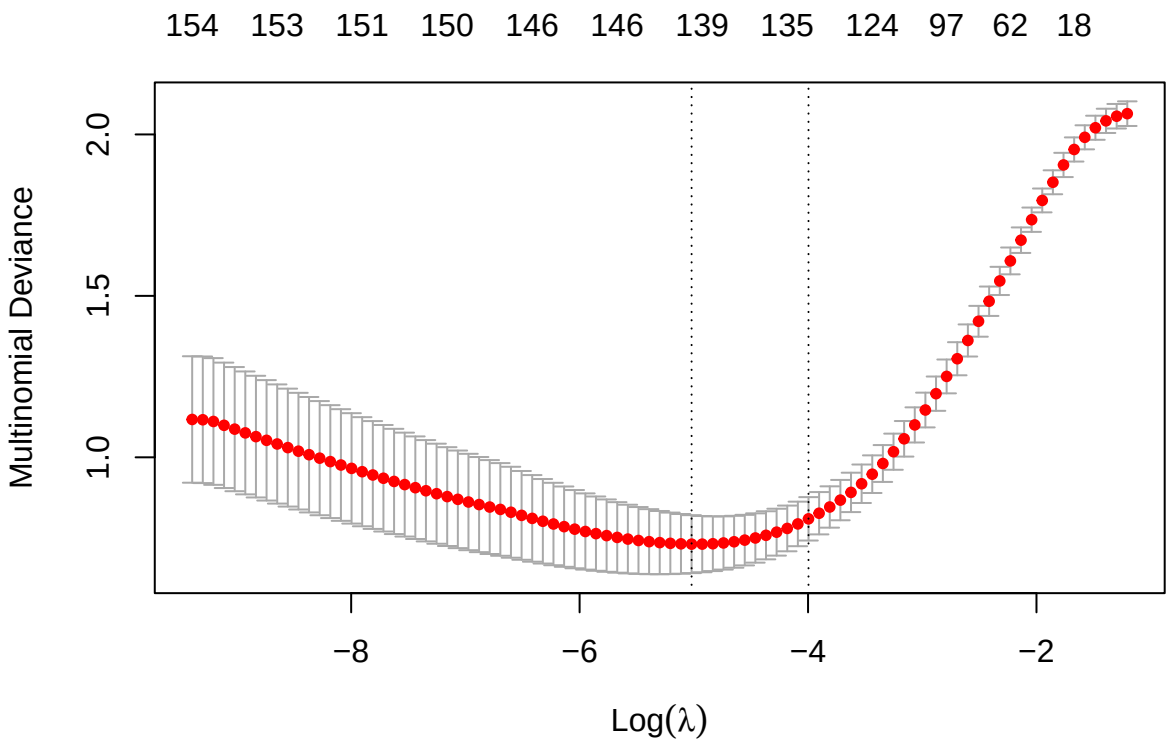



```
# balance weighted
wm_mn_nw <- mn_reg(pca_wm, weight=FALSE)
```

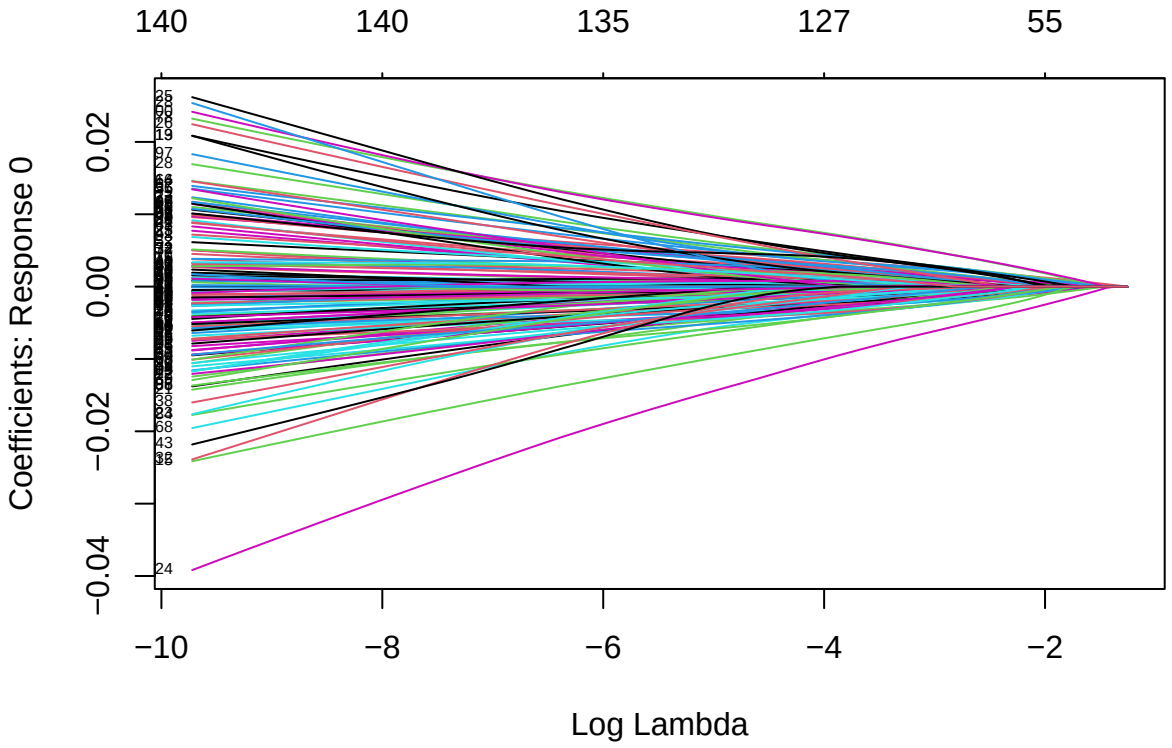


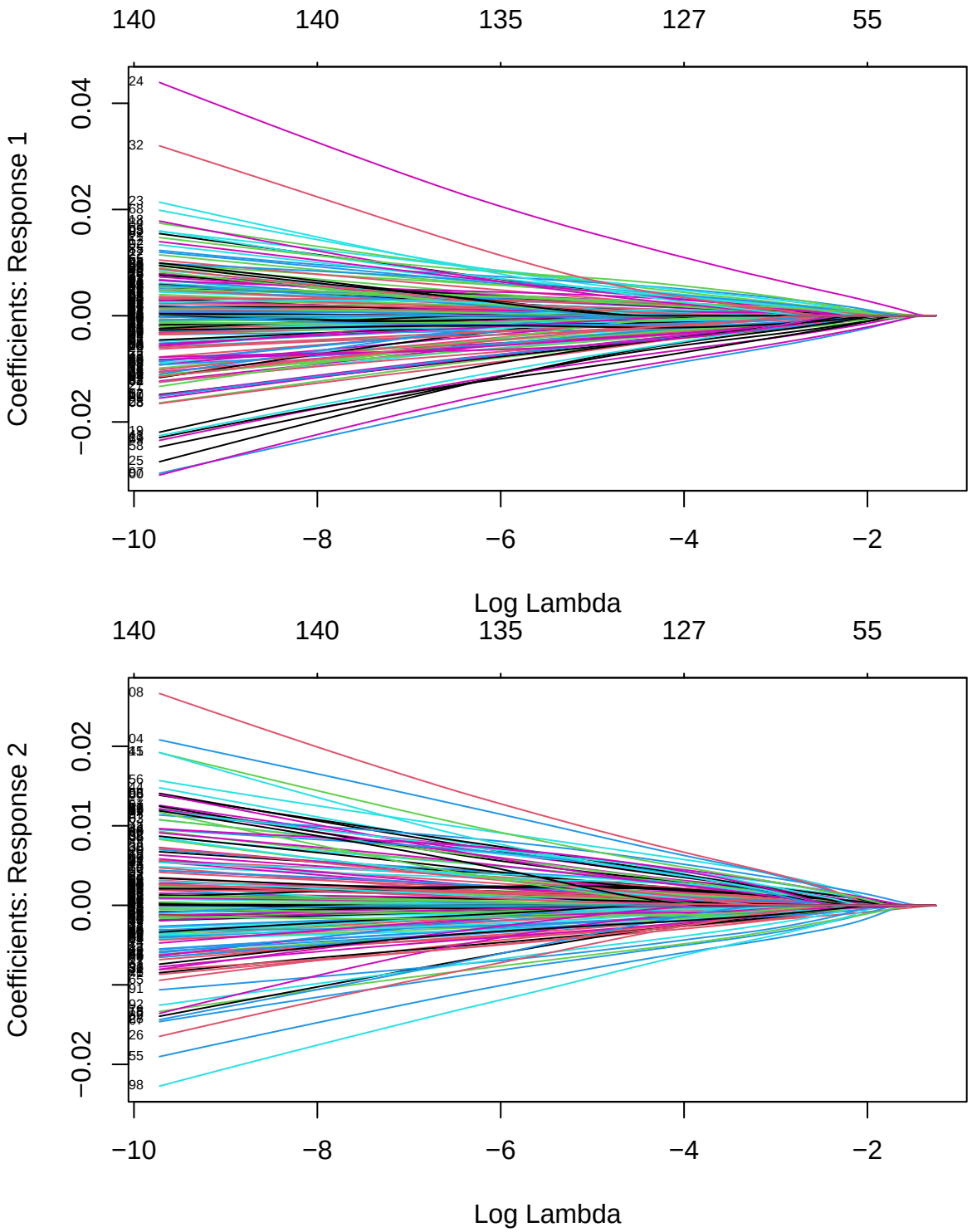



```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

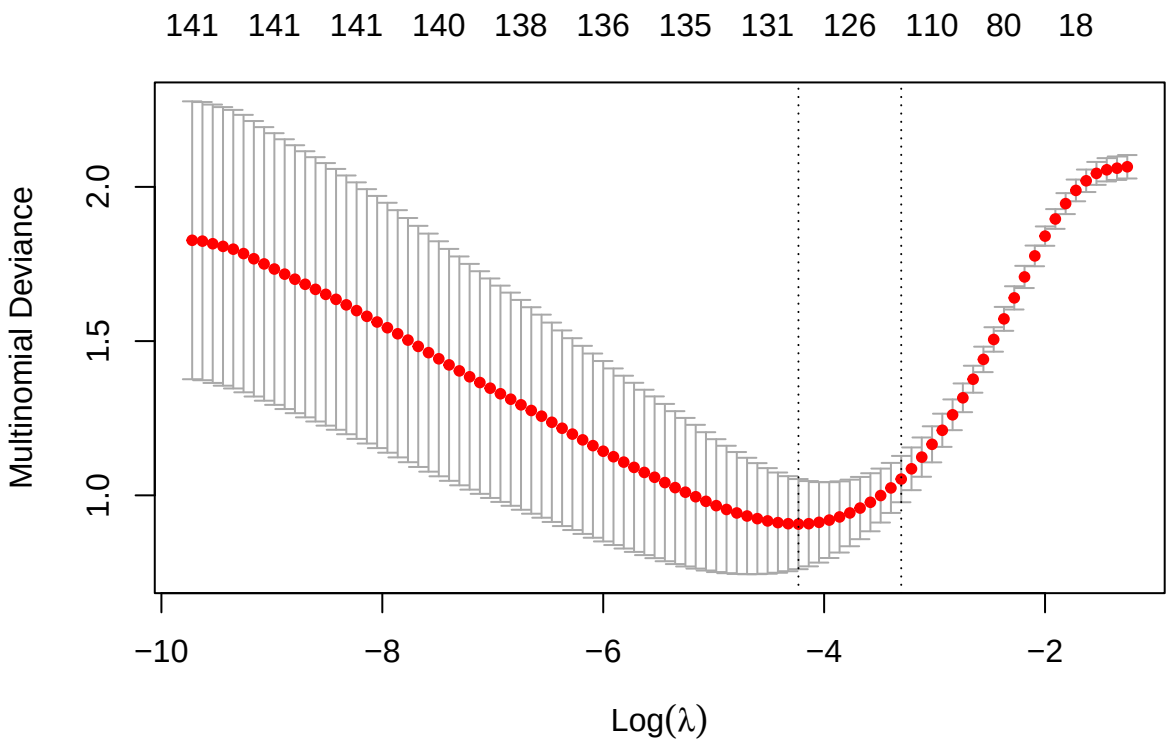



```
gm_mn_nw <- mn_reg(pca_gm, weight=FALSE)
```

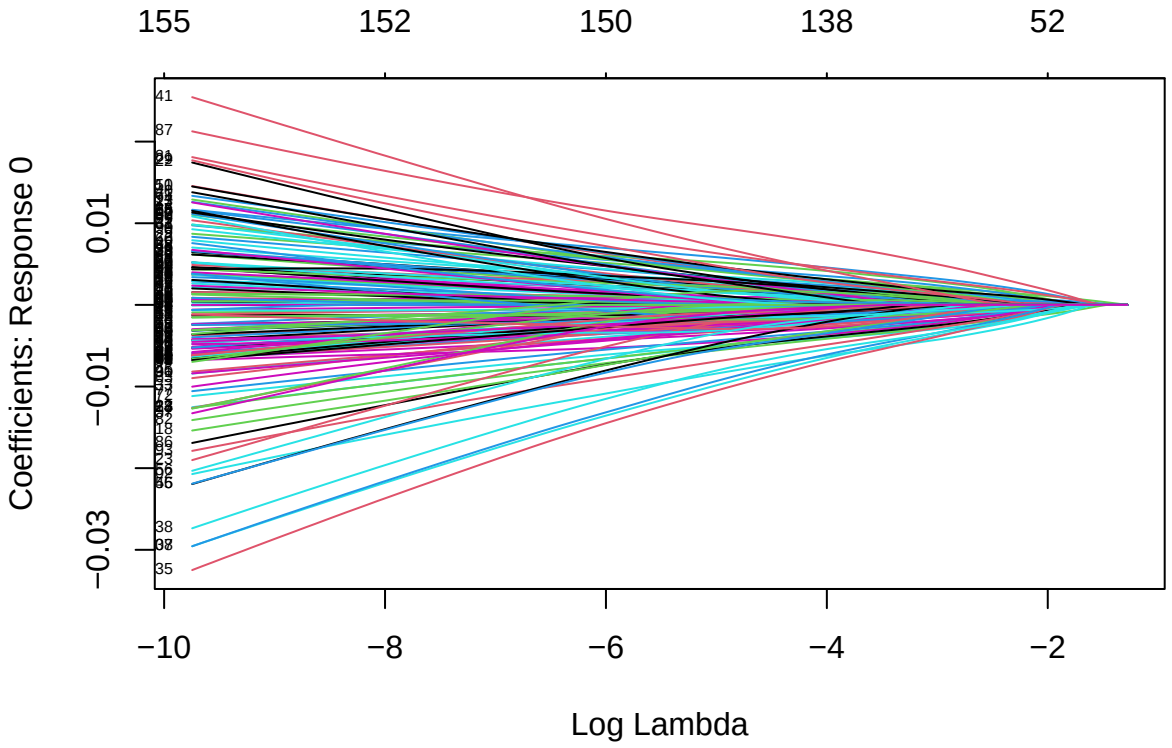


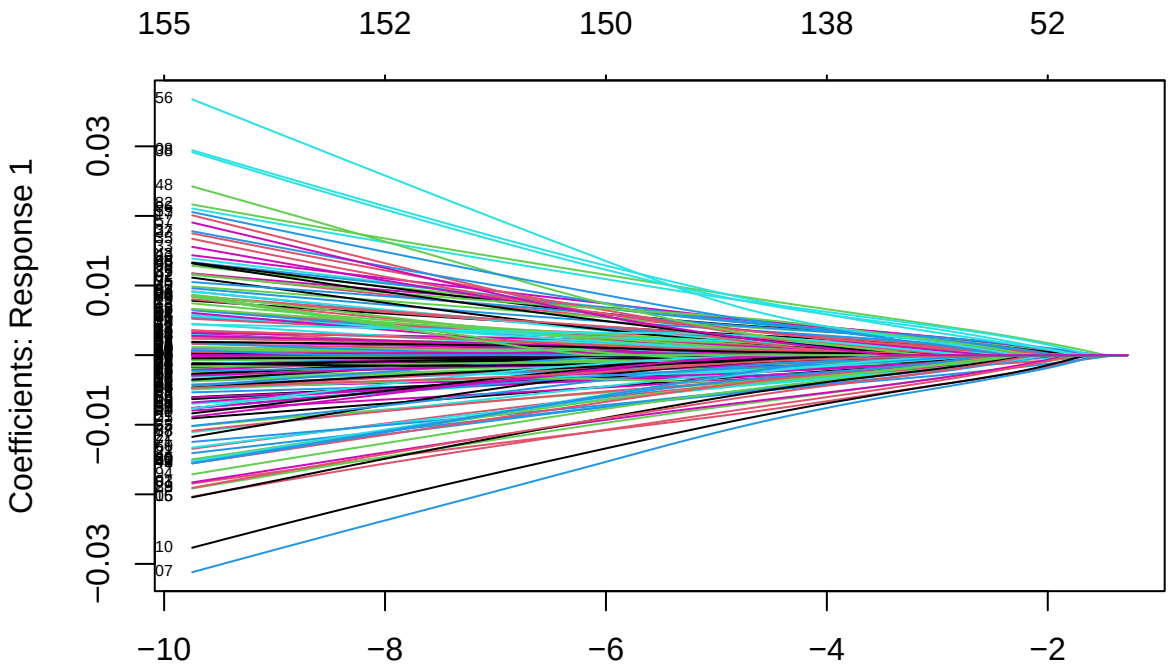


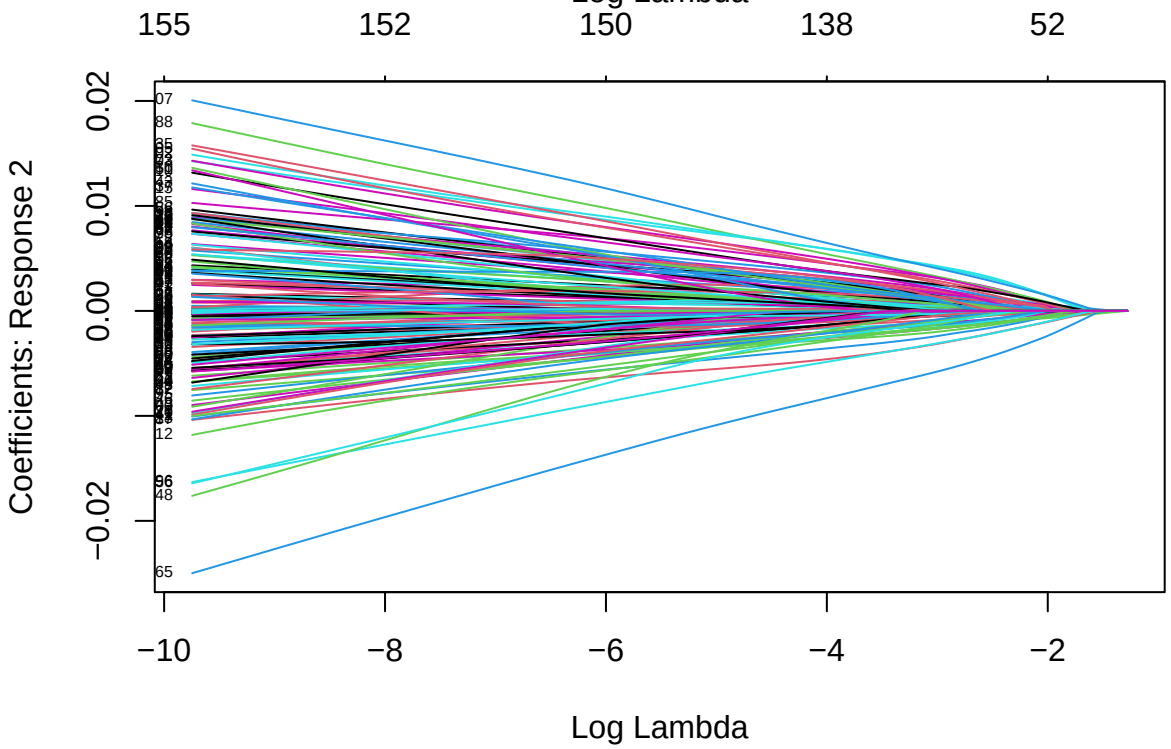

```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```



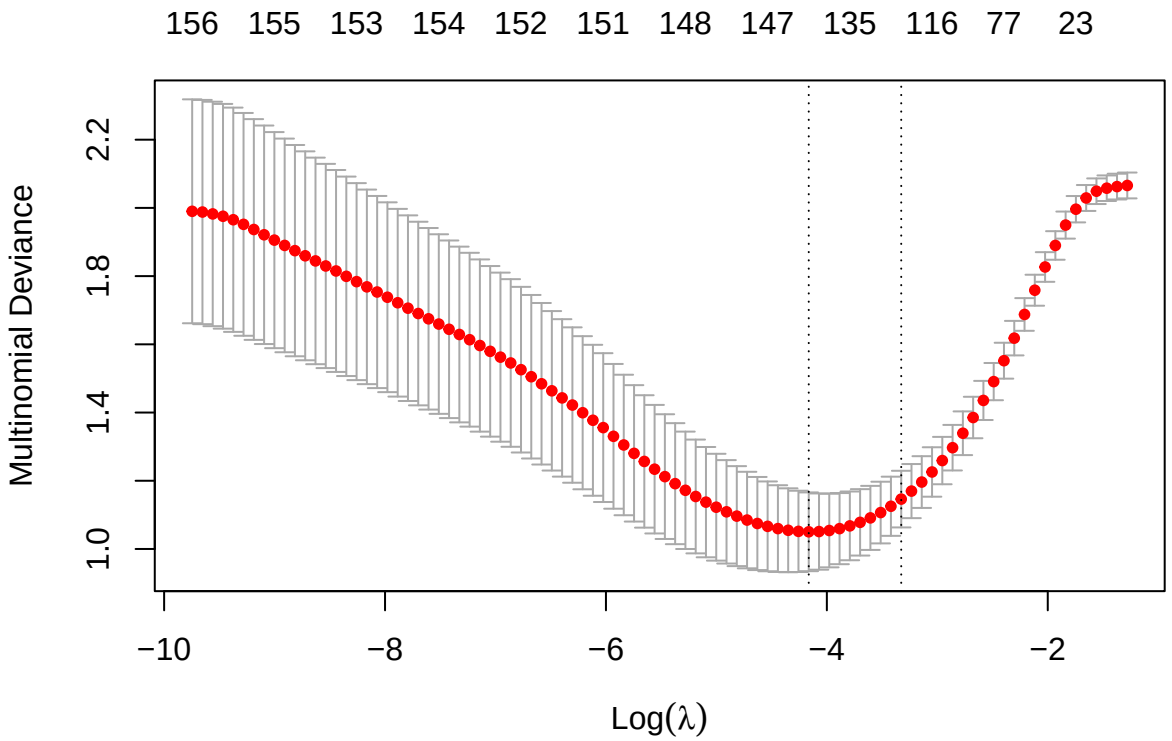

```
cb_mn_nw <- mn_reg(pca_cb, weight=FALSE)
```








```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```



Coefficient Log(lambda) plots This is just like l1 norm (Lasso) where as you increase your penalty, you can see the coefficients shrinking to zero.

https://stats.stackexchange.com/a/186907

Explanation of Deviance Log(lambda) plots Deviance is a specific transformation of a likelihood ratio. In particular, we consider the model-based likelihood after some fitting has been done and compare this to the likelihood of what is called the saturated model. This latter is a model that has as many parameters as data points and achieves a perfect fit, so by looking at the likelihood ratio we're measuring in some sense how far our fitted model is from a "perfect" model.

here we can see the cross validated fit for each log(lambda) you can see the upper and lower standard deviations with the points the first line is the lambda min that gives the minimum mean cross-validated misclassificaiton error the one to the right is the value of lambda that gives the most regularized model such that the cross-validated error is within one standard error of the minimum.

Summaries

If we look at the coefficient section, we're seeing two models, one modelling the effect of principal components 1:n on the log odds of predicting MCI over CN, while the next model is predicting AD over CN. In multinomial logistic regression, we are estimating coefficients for each level of the response variable (MCI, AD, or CN) relative to a baseline level (CN).

We can interpret the coefficients as say for PC1 and 1. Each additional unit of PC1 decreases the log odds of predicting MCI over CN by 0.001449435. This doesn't tell you much, but its good practice to try and interpret.

confusion matrix

```
wm_mn$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 27  0  0
##          1  2  5  1
##          2  1  0  1
```

```
gm_mn$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 29  0  0
##          1  0  5  0
##          2  1  0  2
```

```
cb_mn$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 30  0  0
##          1  0  5  0
##          2  0  0  2
```

```
wm_mn_nw$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 27  0  0
##          1  2  5  1
##          2  1  0  1
```

```
gm_mn_nw$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 30  0  0
##          1  0  5  0
##          2  0  0  2
```

```
cb_mn_nw$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 28  0  0
##          1  2  5  1
##          2  0  0  1
```

One problem with accuracy is that if we have unbalanced data, it can be deceiving. Imagine you have two classes apples (99% of the data) and oranges (1% of the data). We might have a 99% accuracy even though we wrongly predicted the orange. This would be wrong. In our case, we have examples like with our white matter multinomial model where we have an 86% accuracy, but if we split it up by each group, the accracy is high because we correctly classified 40/41 MCI. However, we only correctly classified AD 8/16.

Initially, we faced challenges in getting accurate results due to unbalanced splits, as our test data would sometimes consist mostly of MCI and CN samples, with very few AD samples. To solve this we split the data proportionally and got better results like the above.

To explain, in our full dataset proportionally there are 151/434 (0.348) CN, 206/434 (.475) MCI, and 77/434 (0.177) AD. Originally we then did an 80/20 split, where 80% or 347 of the 434 samples are set aside for the train and 87 are set aside for the test. However, this time, of those 347 sample, 34.8% will be CN, 47.5% will

be MCI and 17.7% will be AD.

This misclassification can cause further problems as having a false negative (failing to classifying someone as AD and instead saying theyre CN) can be more fatal and thus should have a higher weight.

LDA

```
wm_lda <- lda_reg(pca_wm)
```

```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

```
gm_lda <- lda_reg(pca_gm)
```

```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

```
cb_lda <- lda_reg(pca_cb)
```

```
## Warning in cohen.kappa1(x, w = w, n.obs = n.obs, alpha = alpha, levels =
## levels): upper or lower confidence interval exceed abs(1) and set to +/- 1.
```

```
wm_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 27  0  0
##          1  2  5  1
##          2  1  0  1
```

```
gm_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 29  0  0
##          1  0  5  0
##          2  1  0  2
```

```
cb_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##          0 29  0  0
##          1  0  5  0
##          2  1  0  2
```

LDA Assumptions

Homoskedasticity Among Classes Bartlett's K-Squared Test

$H_0 : \sigma_{CN} = \sigma_{MCI} = \sigma_{AD}$ Variance is the same across all classes

$H_1 : \sigma_{CN} \neq \sigma_{MCI} \neq \sigma_{AD}$ Variance isn't the same across all classes

$\alpha = 0.05$

```
## Bart Test
b_test_wm <- bart_test(pca_wm)
```

```
##
##  Bartlett test of homogeneity of variances
```

```
##
## data:  var_df$Var and var_df$Group
## Bartlett's K-squared = 24.979, df = 2, p-value = 3.766e-06
##
## [1] "We REJECT the null that the variances are the same across all classes"
```

```
head(b_test_wm$var_df)
```

```
## # A tibble: 6 x 2
##   Group     Var
##   <chr>   <dbl>
## 1 CN    233160.
## 2 MCI   169768.
## 3 AD    234514.
## 4 CN     82975.
## 5 MCI   100754.
## 6 AD     63098.
```

```
class_data_wm <- b_test_wm$class_data

##
b_test_gm <- bart_test(pca_gm)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  var_df$Var and var_df$Group
## Bartlett's K-squared = 47.148, df = 2, p-value = 5.781e-11
##
## [1] "We REJECT the null that the variances are the same across all classes"
```

```
head(b_test_gm$var_df)
```

```
## # A tibble: 6 x 2
##   Group     Var
##   <chr>   <dbl>
## 1 CN    272023.
## 2 MCI   165765.
## 3 AD    339993.
## 4 CN    124614.
## 5 MCI    87270.
## 6 AD     77031.
```

```
class_data_gm <- b_test_gm$class_data_wm

##
b_test_cb <- bart_test(pca_cb)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  var_df$Var and var_df$Group
## Bartlett's K-squared = 45.119, df = 2, p-value = 1.594e-10
##
## [1] "We REJECT the null that the variances are the same across all classes"
```

```
head(b_test_cb$var_df)
```

```
## # A tibble: 6 x 2
##   Group     Var
##   <chr>   <dbl>
## 1 CN    264715.
## 2 MCI   166266.
## 3 AD    331813.
## 4 CN    119378.
## 5 MCI    82605.
## 6 AD     67996.
```

```
class_data_cb <- b_test_cb$class_data_gm
```

Normality Among Classes Kolmogorov-Smirnov Normality Test

$H_0 : X_{ik} \sim \text{Normal}(\mu, \sigma) \forall i \in 1, ..., n, k \in 0, 1, 2$ Feature follows a normal distribution

H_1 : Feature does not follow a normal distribution

$\alpha = 0.05$

```
## KS Test
k_test_wm <- ks_test(pca_wm)

head(k_test_wm$ks_df)
```

```
##     CN MCI           AD
## PC1  0   0 1.665335e-15
## PC2  0   0 2.442491e-15
## PC3  0   0 1.665335e-15
## PC4  0   0 1.665335e-15
## PC5  0   0 1.665335e-15
## PC6  0   0 1.665335e-15
```

```
head(k_test_wm$normal_pcs)
```

```
## [1] CN  MCI AD
## <0 rows> (or 0-length row.names)
```

```
k_test_wm$non_normal_pcs %>%
    summarise(non_normal_count = n())
```

```
##   non_normal_count
## 1              160
```

```
k_test_gm <- ks_test(pca_gm)

head(k_test_gm$ks_df)
```

```
##     CN MCI           AD
## PC1  0   0 1.665335e-15
## PC2  0   0 1.665335e-15
## PC3  0   0 1.665335e-15
## PC4  0   0 1.665335e-15
## PC5  0   0 1.665335e-15
## PC6  0   0 1.665335e-15
```

```
head(k_test_gm$normal_pcs)
```

```
## [1] CN  MCI AD
## <0 rows> (or 0-length row.names)
```

```
k_test_gm$non_normal_pcs %>%
    summarise(non_normal_count = n())
```

```
##   non_normal_count
## 1              144
```

```
k_test_cb <- ks_test(pca_cb)

head(k_test_cb$ks_df)
```

```
##     CN MCI           AD
## PC1  0   0 1.665335e-15
## PC2  0   0 1.665335e-15
## PC3  0   0 4.884981e-15
## PC4  0   0 1.665335e-15
## PC5  0   0 1.665335e-15
## PC6  0   0 1.665335e-15
```

```
head(k_test_cb$normal_pcs)
```

```
## [1] CN  MCI AD
## <0 rows> (or 0-length row.names)
```

```
k_test_cb$non_normal_pcs %>%
    summarise(non_normal_count = n())
```

```
##   non_normal_count
## 1              157
```

We reject the null for each feature for all classes. LDA assumptions are not met.

Precision for class i: TP_i / (TP_i + FP_i)

Recall for class i: TP_i / (TP_i + FN_i)

TP_i = True Positives for class i (number of instances correctly classified as class i)

FP_i = False Positives for class i (number of instances incorrectly classified as class i)

FN_i = False Negatives for class i (number of instances incorrectly classified as not class i, but actually belong to class i)

Method comparison

```
df <- data.frame(rbind(get_other_scores_table(wm_mn$scores),
                get_other_scores_table(gm_mn$scores),
                get_other_scores_table(cb_mn$scores)))

rownames(df) <- NULL
df <- df %>%
    mutate(model = c('White matter', rep('',2),
                     'Gray matter', rep('',2),
                     'Combined matter', rep('',2)),
      score = rep(c('precision', 'recall', 'f1'), 3)) %>%
```

```
    select(model, score, everything())
kable(df, caption='Multinomial Logistic Regression Scores')
```

Table 1: Multinomial Logistic Regression Scores

model	score	CN	MCI	AD
White matter	precision	1.0000000	0.6250000	0.5000000
	recall	0.9000000	1.0000000	0.5000000
	f1	0.9473684	0.7692308	0.5000000
Gray matter	precision	1.0000000	1.0000000	0.6666667
	recall	0.9666667	1.0000000	1.0000000
	f1	0.9830508	1.0000000	0.8000000
Combined matter	precision	1.0000000	1.0000000	1.0000000
	recall	1.0000000	1.0000000	1.0000000
	f1	1.0000000	1.0000000	1.0000000

```
df <- data.frame(rbind(get_other_scores_table(wm_lda$scores),
                get_other_scores_table(gm_lda$scores),
                get_other_scores_table(cb_lda$scores)))

rownames(df) <- NULL
df <- df %>%
    mutate(model = c('White matter', rep('',2),
                     'Gray matter', rep('',2),
                     'Combined matter', rep('',2)),
      score = rep(c('precision', 'recall', 'f1'), 3)) %>%
    select(model, score, everything())
kable(df, caption='LDA Scores')
```

Table 2: LDA Scores

model	score	CN	MCI	AD
White matter	precision	1.0000000	0.6250000	0.5000000
	recall	0.9000000	1.0000000	0.5000000
	f1	0.9473684	0.7692308	0.5000000
Gray matter	precision	1.0000000	1.0000000	0.6666667
	recall	0.9666667	1.0000000	1.0000000
	f1	0.9830508	1.0000000	0.8000000
Combined matter	precision	1.0000000	1.0000000	0.6666667
	recall	0.9666667	1.0000000	1.0000000
	f1	0.9830508	1.0000000	0.8000000

```
df <- data.frame(model = c('Multinomial', rep('',2),
                           'Multinomial nw', rep('',2),
                           'LDA', rep('',2)),
           plane = rep(c('White matter', 'Gray matter', 'Combined matter'), 3),
           rbind(wm_mn$scores[[4]],
                 gm_mn$scores[[4]],
                 cb_mn$scores[[4]],
                 wm_mn_nw$scores[[4]],
                 gm_mn_nw$scores[[4]],
```

```
                cb_mn_nw$scores[[4]],
                wm_lda$scores[[4]],
                gm_lda$scores[[4]],
                cb_lda$scores[[4]]
               ))

kable(df, caption='Accuracies across models')
```

Table 3: Accuracies across models

model	plane	acc	mce
Multinomial	White matter	0.8918919	0.1081081
	Gray matter	0.972973	0.02702703
	Combined matter	1	0
Multinomial nw	White matter	0.8918919	0.1081081
	Gray matter	1	0
	Combined matter	0.9189189	0.08108108
LDA	White matter	0.8918919	0.1081081
	Gray matter	0.972973	0.02702703
	Combined matter	0.972973	0.02702703

```
df <- data.frame(model = c('Multinomial', rep('',2),
                           'Multinomial nw', rep('',2),
                           'LDA', rep('',2)),
           plane = rep(c('White matter', 'Gray matter', 'Combined matter'), 3),
           rbind(wm_mn$scores[[5]],
                 gm_mn$scores[[5]],
                 cb_mn$scores[[5]],
                 wm_mn_nw$scores[[5]],
                 gm_mn_nw$scores[[5]],
                 cb_mn_nw$scores[[5]],
                 wm_lda$scores[[5]],
                 gm_lda$scores[[5]],
                 cb_lda$scores[[5]]
                ))

kable(df, caption='Scores across models')
```

Table 4: Scores across models

model	plane	var	lb	ub	est
Multinomial	White matter	0.01557474	0.4680202	0.9572225	0.7126214
	Gray matter	0.006028103	0.7687669	1	0.9209402
	Combined matter	0	1	1	1
Multinomial nw	White matter	0.01557474	0.4680202	0.9572225	0.7126214
	Gray matter	0	1	1	1
	Combined matter	0.01374417	0.5422965	1	0.7720739
LDA	White matter	0.01557474	0.4680202	0.9572225	0.7126214
	Gray matter	0.006028103	0.7687669	1	0.9209402
	Combined matter	0.006028103	0.7687669	1	0.9209402