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 error
```

```
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
```

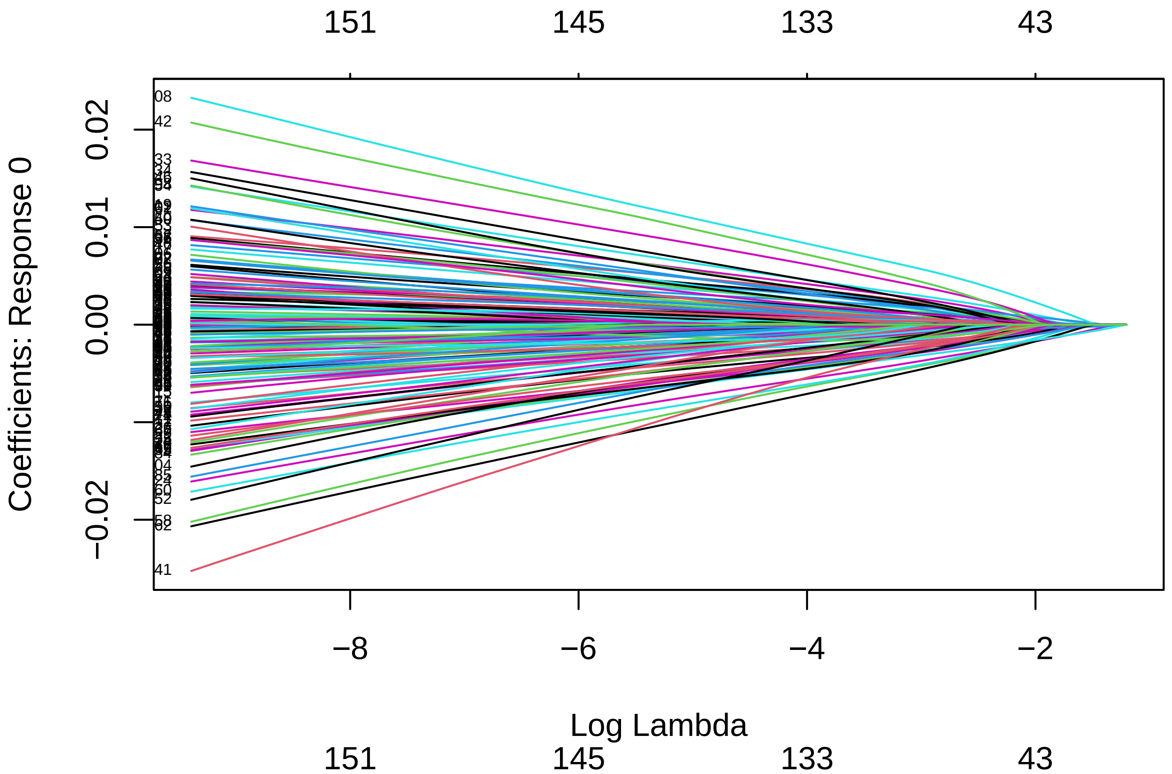
```
library(caret)
```

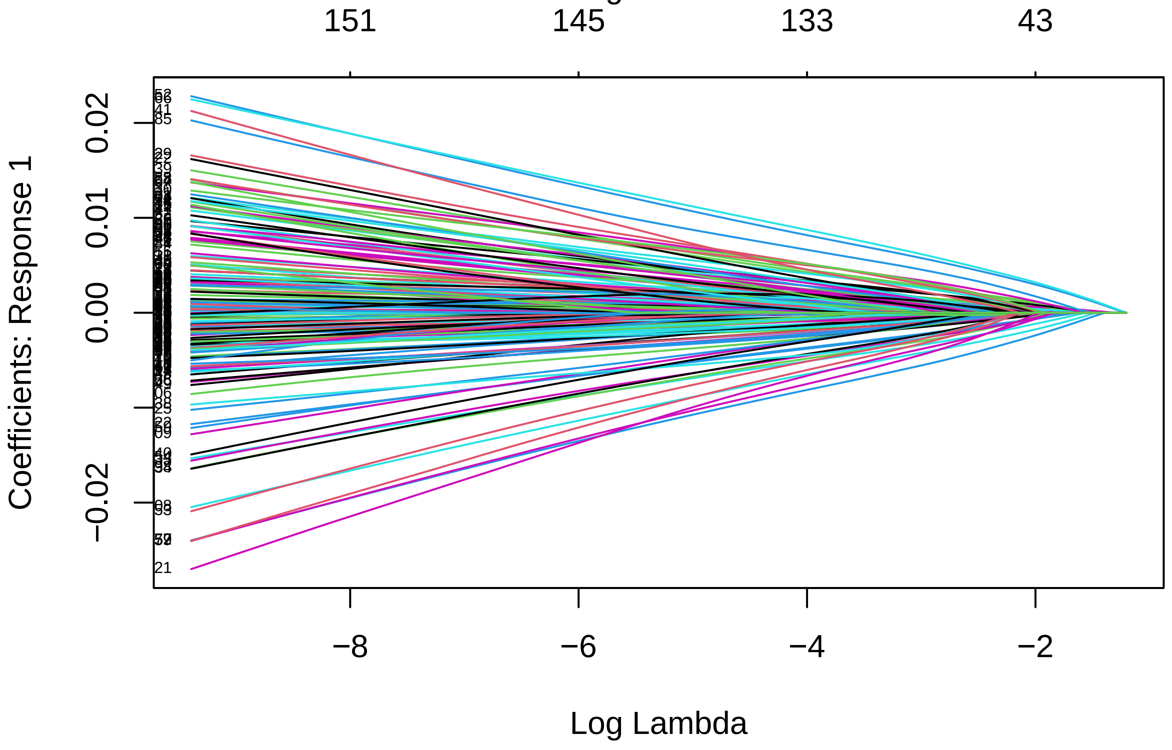
```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

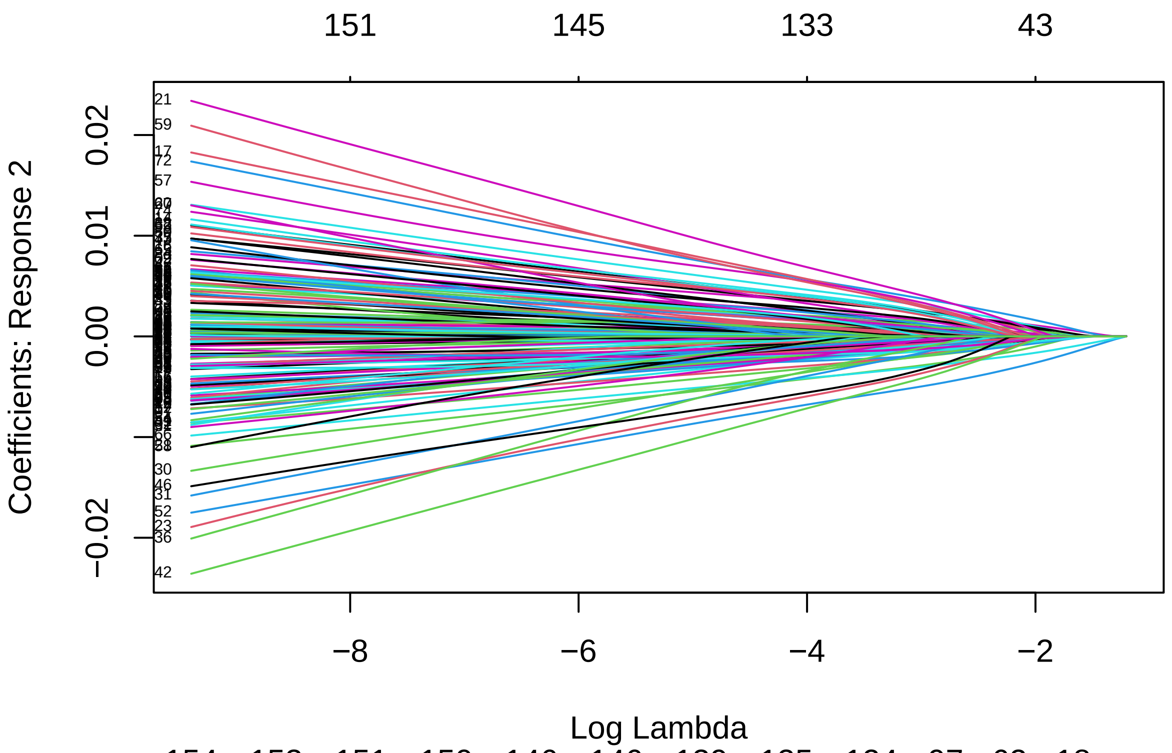
```
load(file = file.path(root_dir, "R_data", "pca_wm.Rda"))
load(file = file.path(root_dir, "R_data", "pca_gm.Rda"))
load(file = file.path(root_dir, "R_data", "pca_cb.Rda"))
```

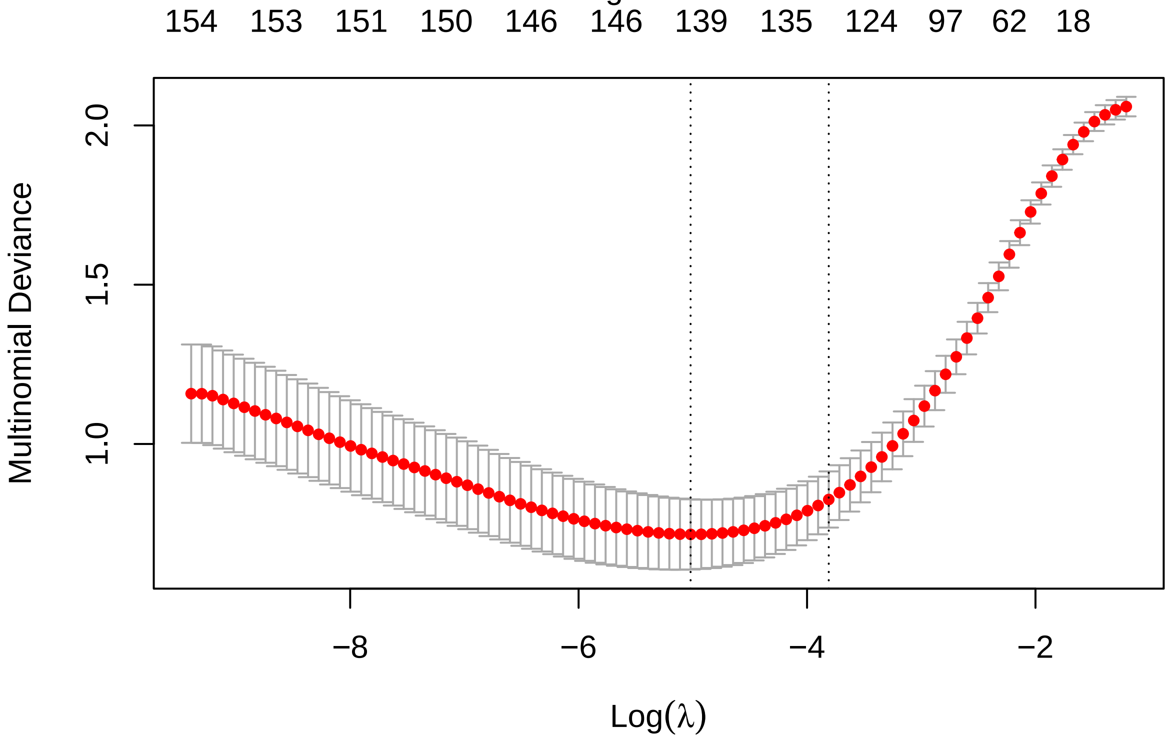
Running models

```
# not balance weighted
wm_mn_nw <- mn_reg(pca_wm)
```

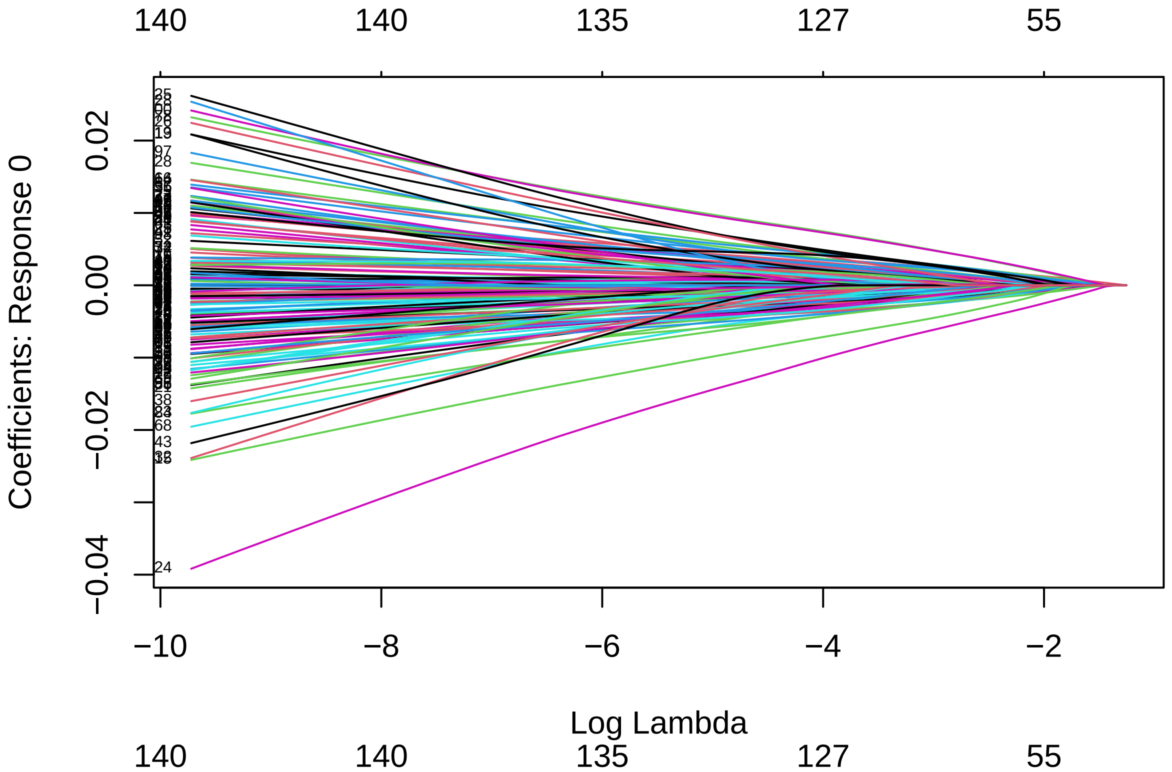
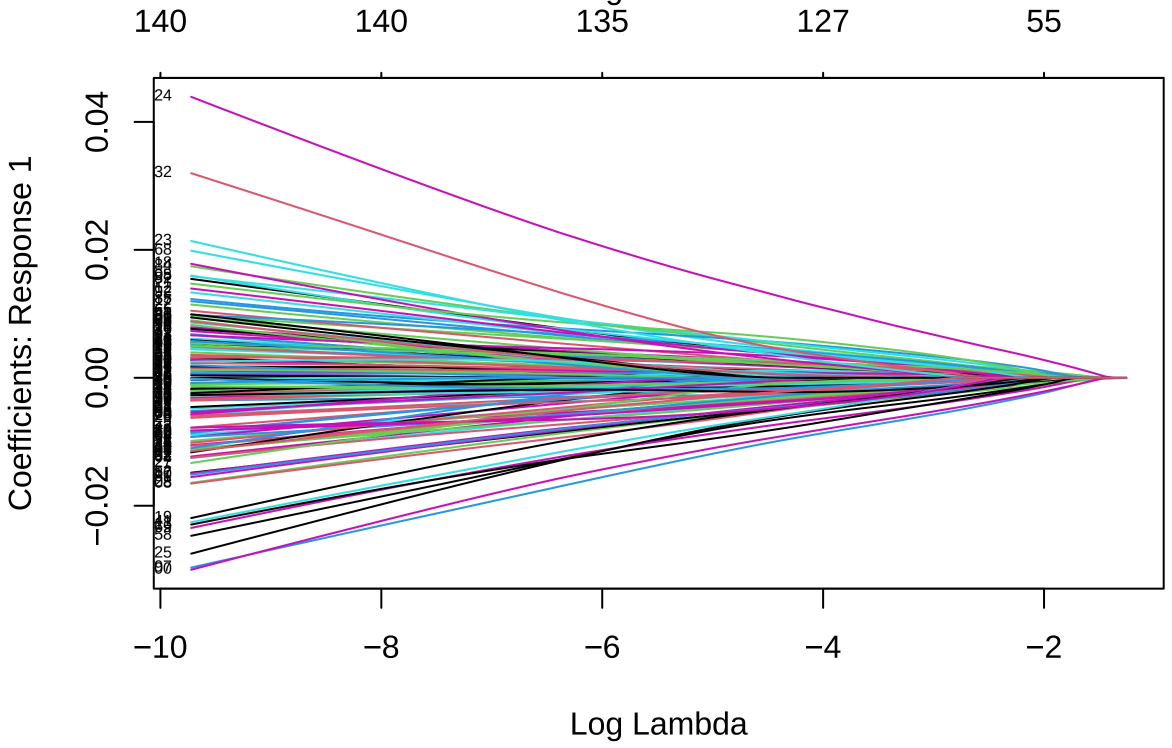


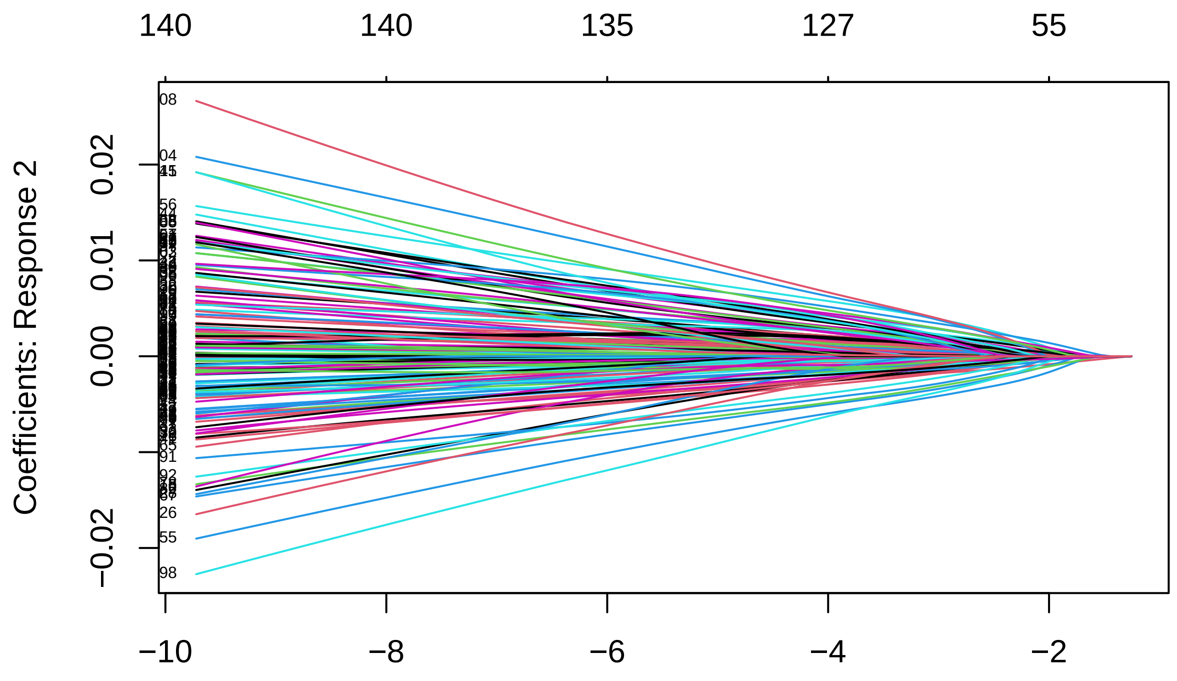


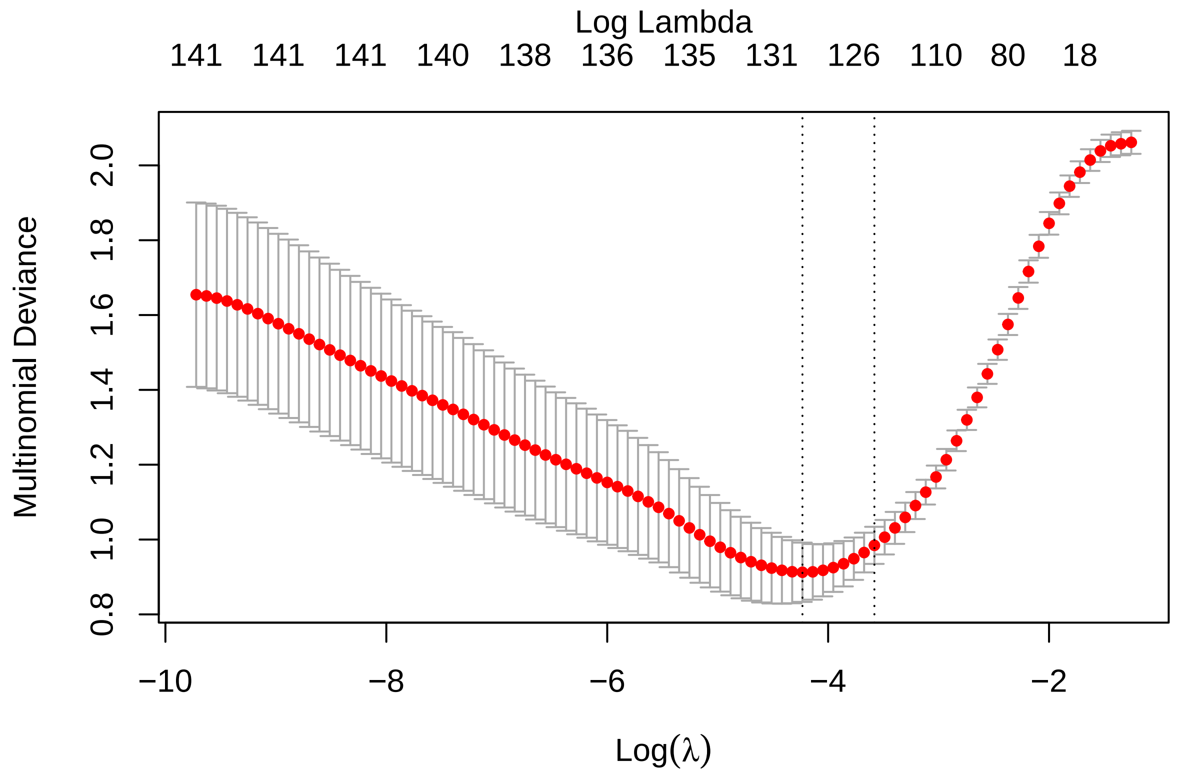




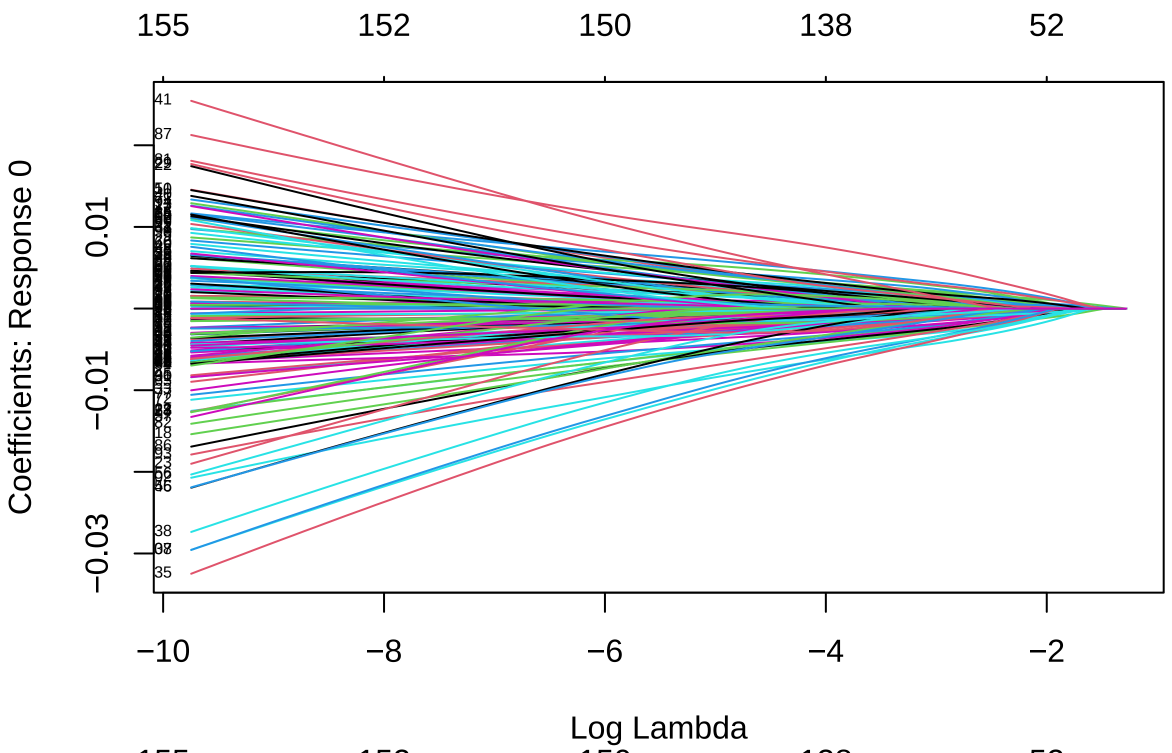
```
gm_mn_nw <- mn_reg(pca_gm)
```

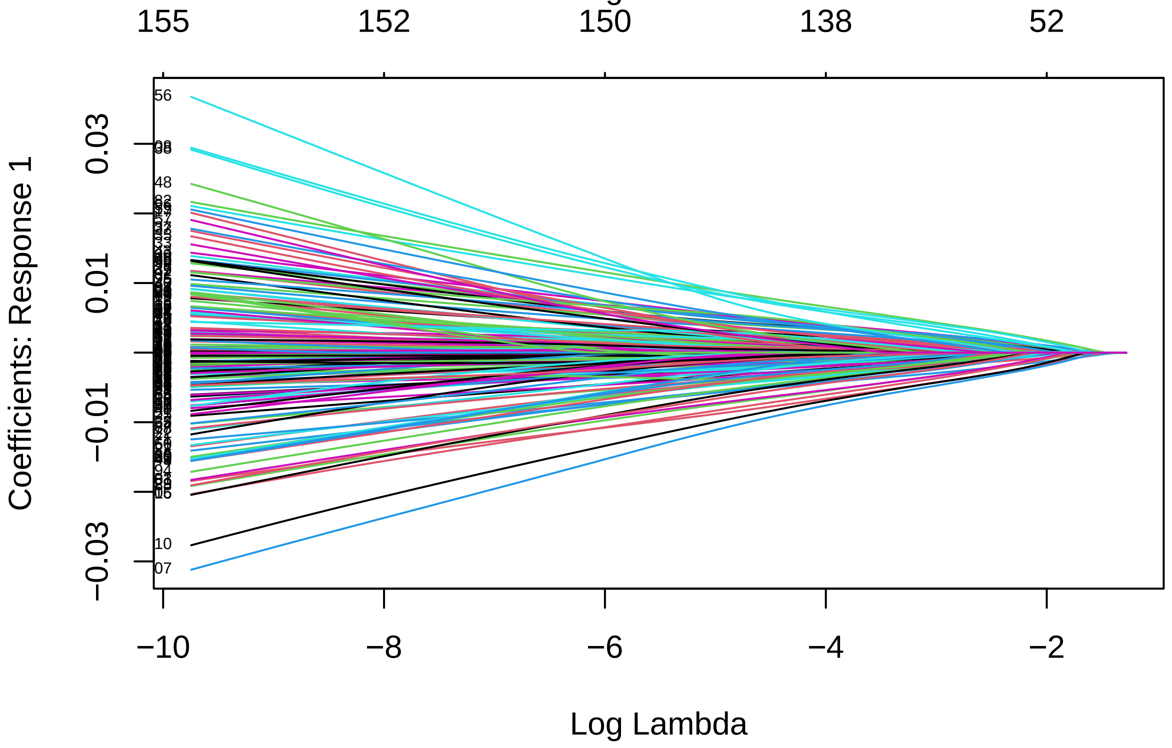

140
140
135
127
55
Coefficients: Response 0
0.02
0.00
−0.02
−0.04
−10
−8
−6
−4
−2
Log Lambda
24
43
68

140
140
135
127
55
Coefficients: Response 1
0.04
0.02
0.00
−0.02
−10
−8
−6
−4
−2
Log Lambda
24
32
23
68
25

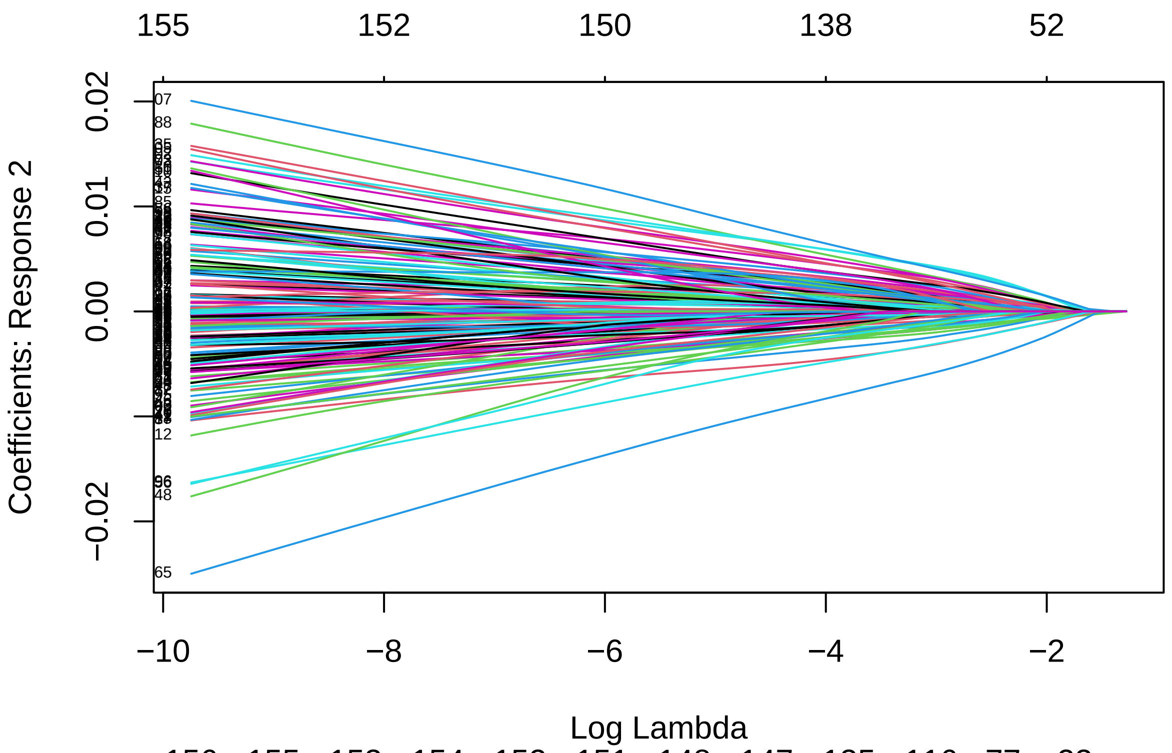


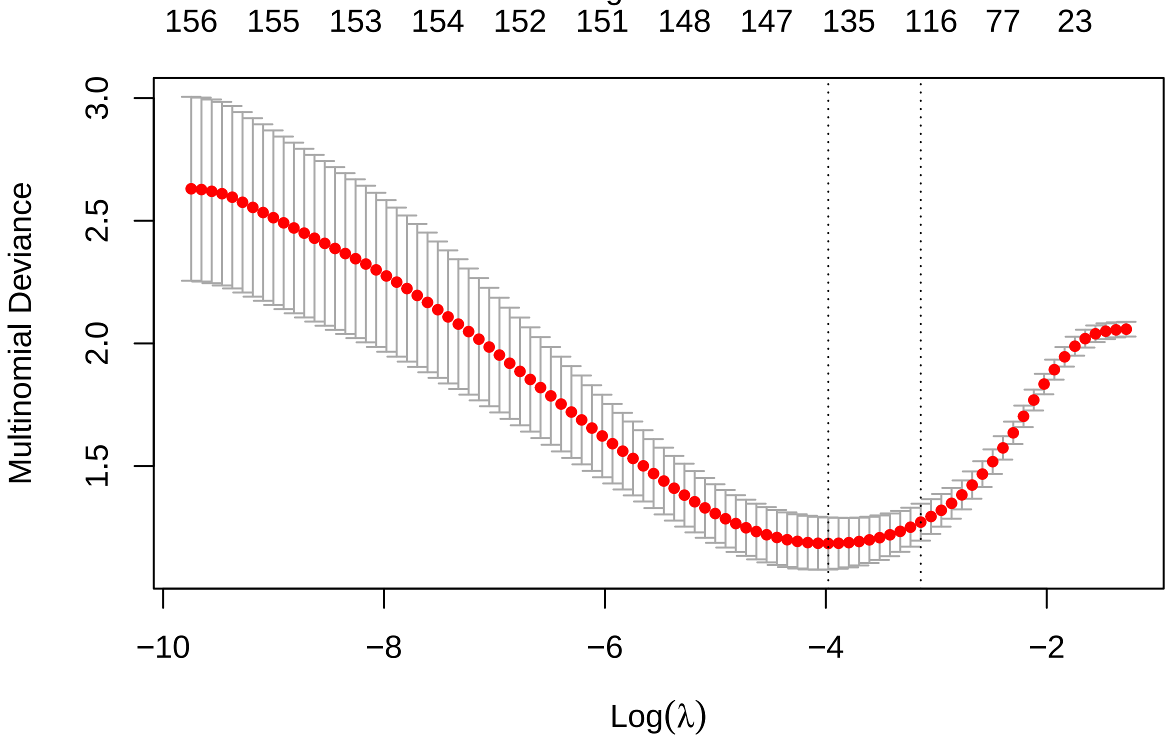


```
cb_mn_nw <- mn_reg(pca_cb)
```

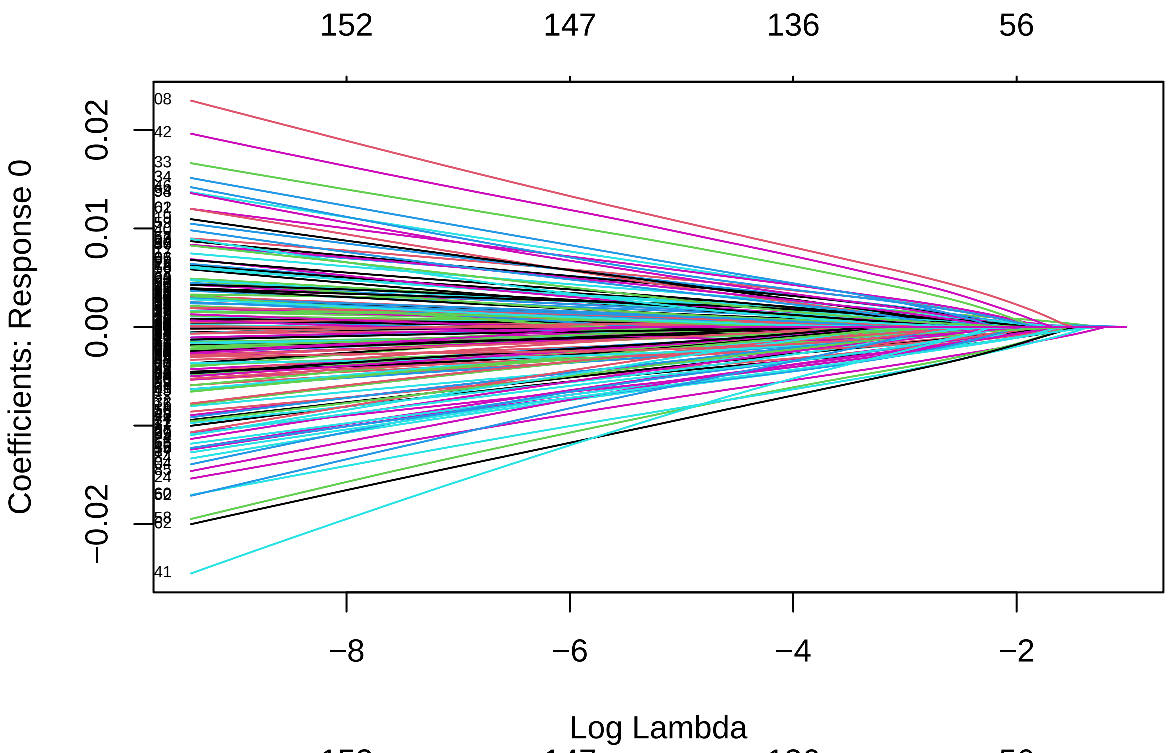

155
152
150
138
52
Coefficients: Response 0
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda
41
87
35
38

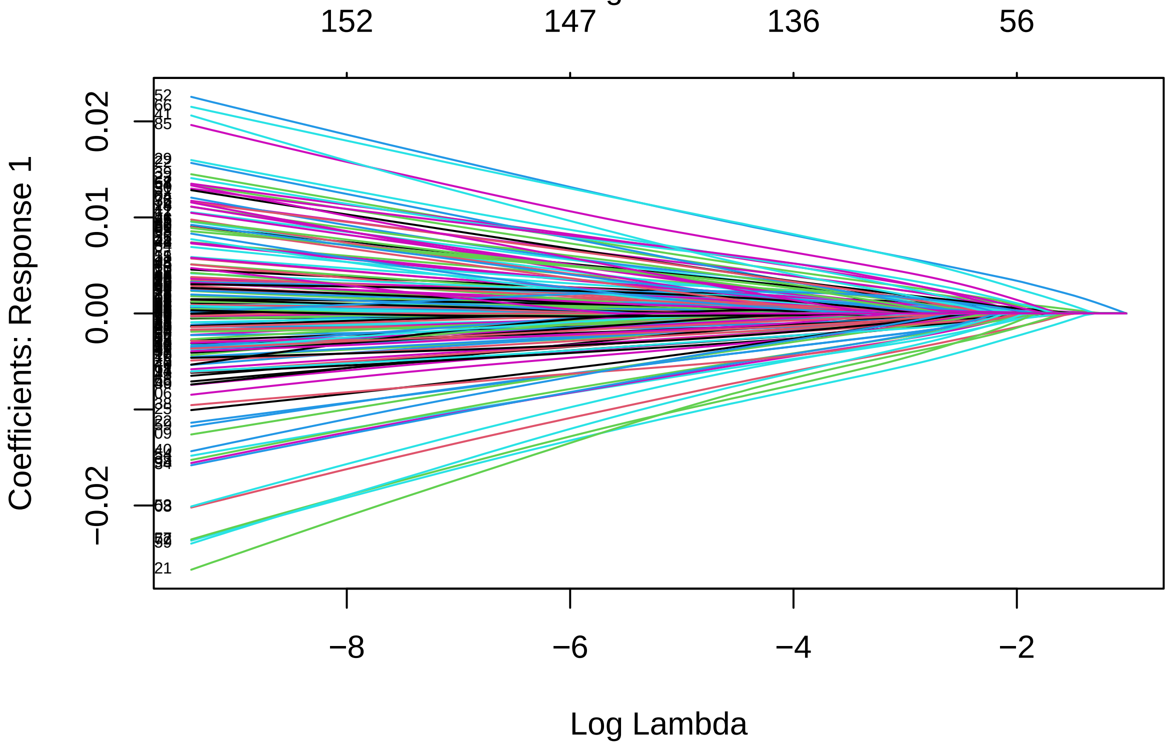

155
152
150
138
52
Coefficients: Response 1
0.03
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda
56
48
10
07

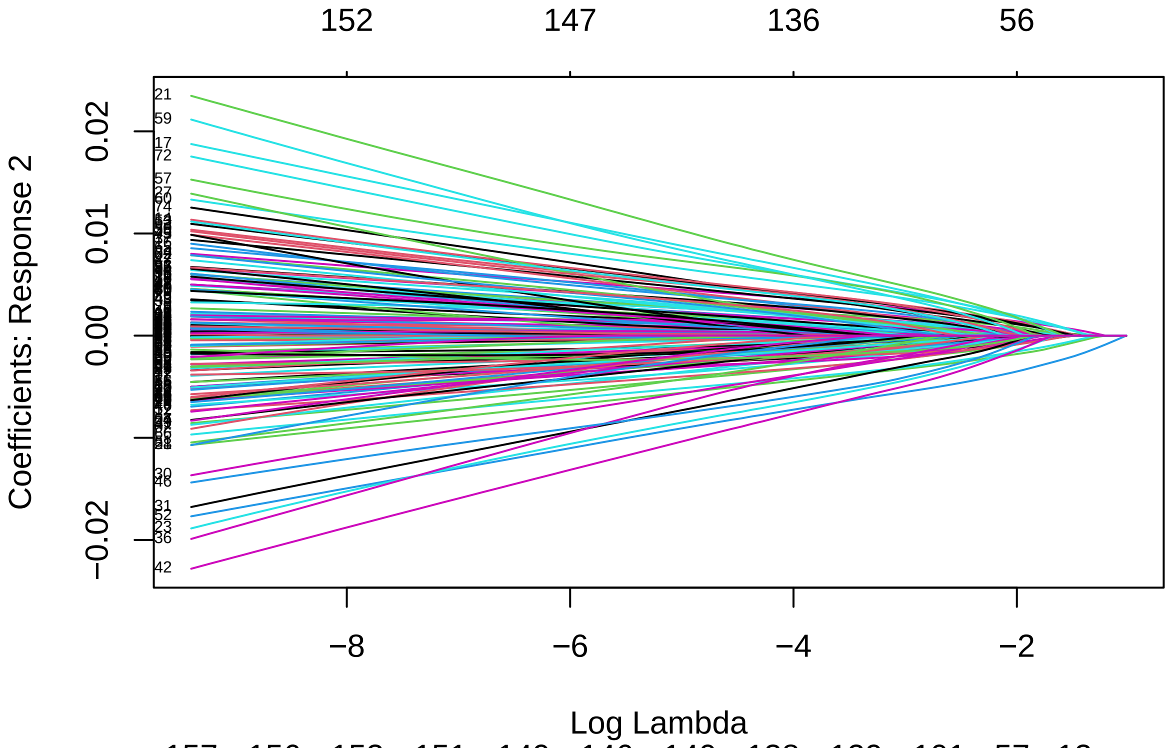


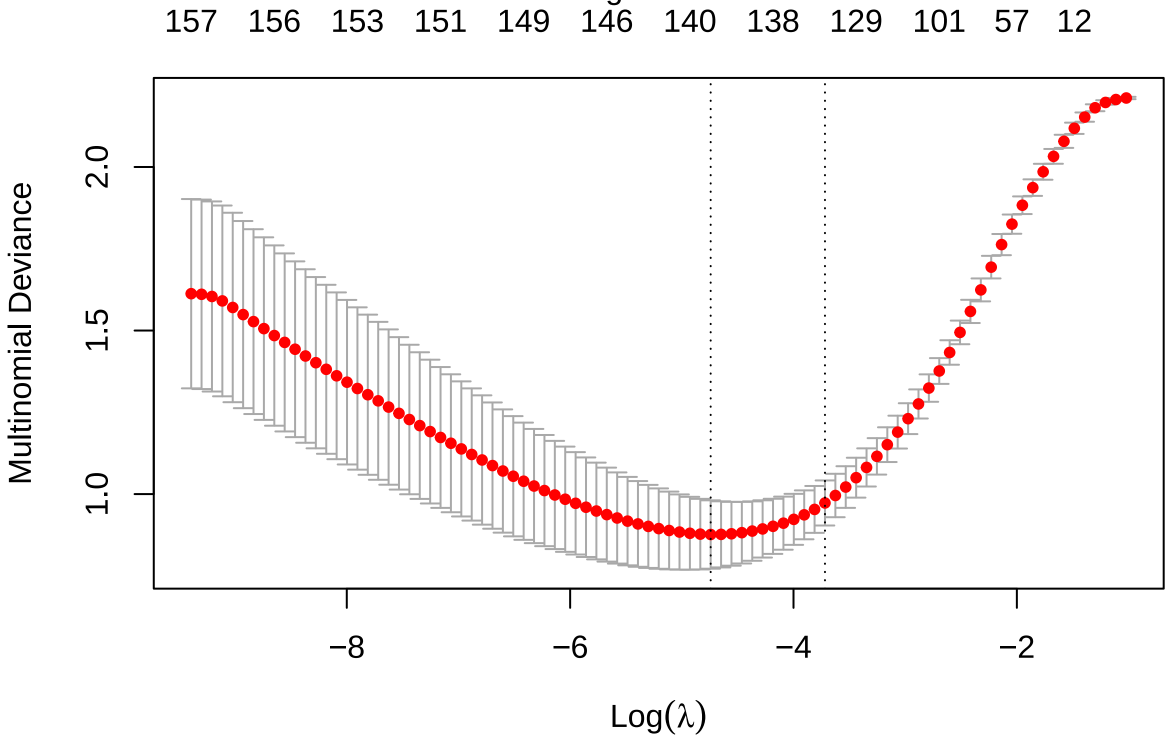


```r
# balance weighted
wm_mn <- mn_reg(pca_wm,weights=TRUE)
```

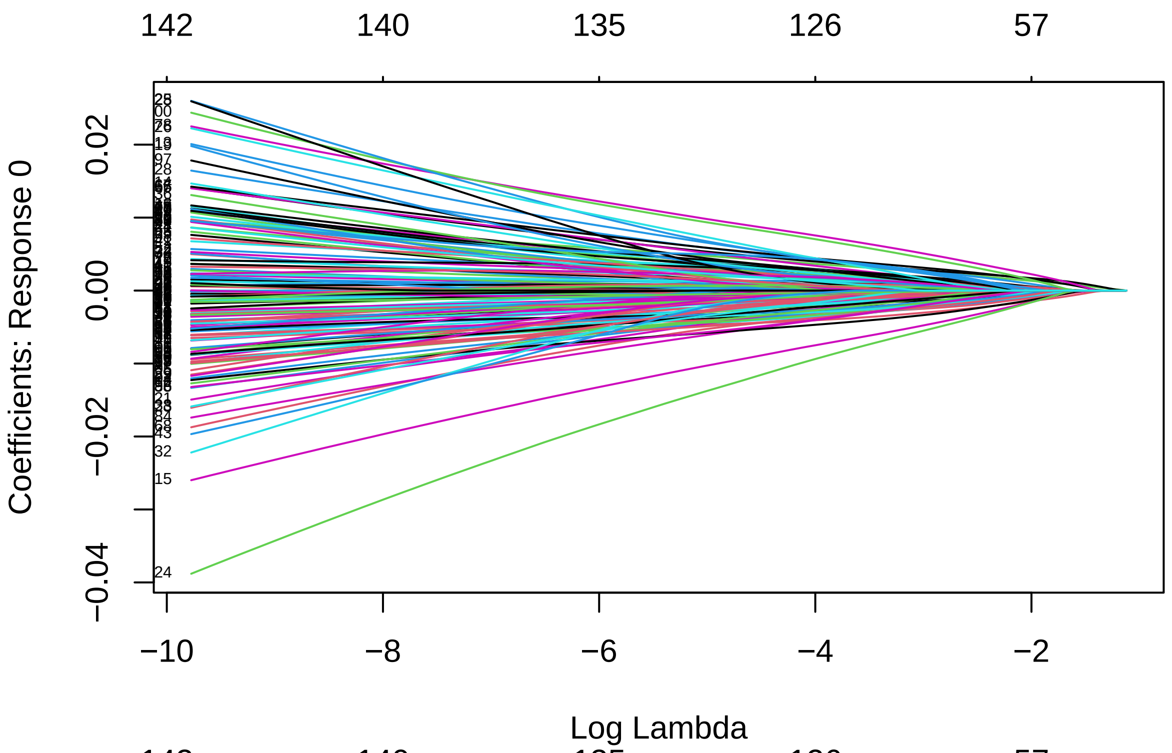

152
147
136
56
Coefficients: Response 0
0.02
0.01
0.00
−0.02
−8
−6
−4
−2
Log Lambda

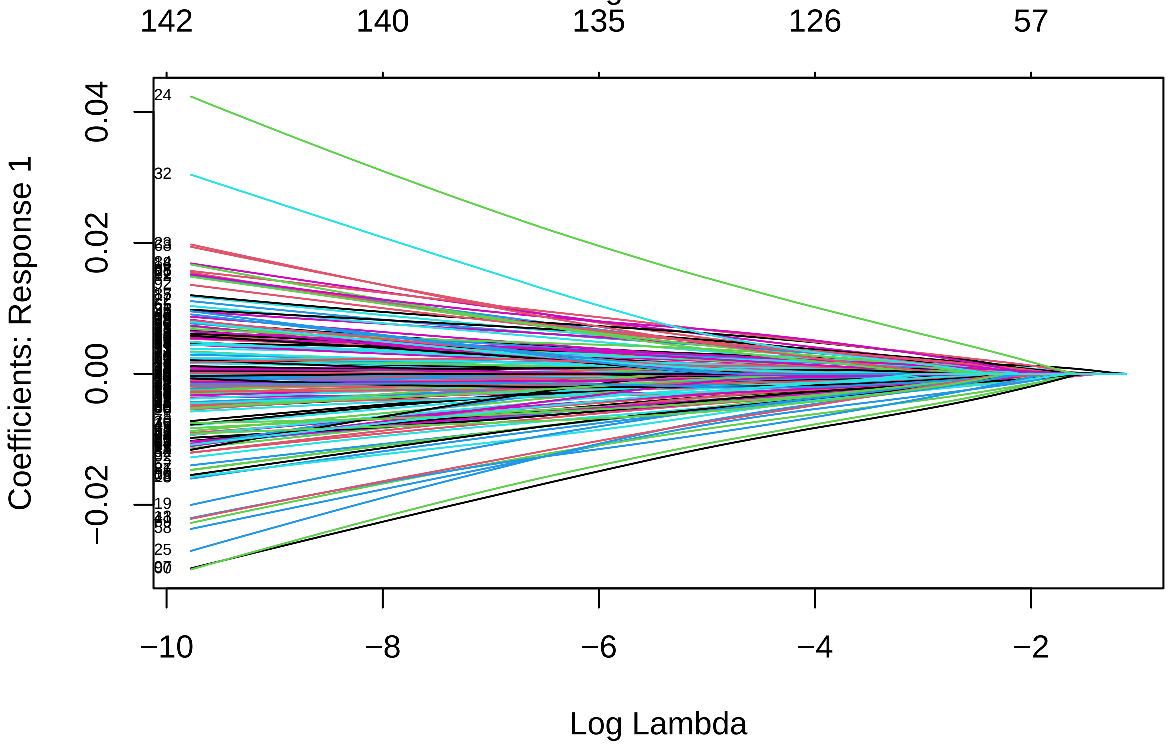

152
147
136
56
Coefficients: Response 1
0.02
0.01
0.00
−0.02
−8
−6
−4
−2
Log Lambda

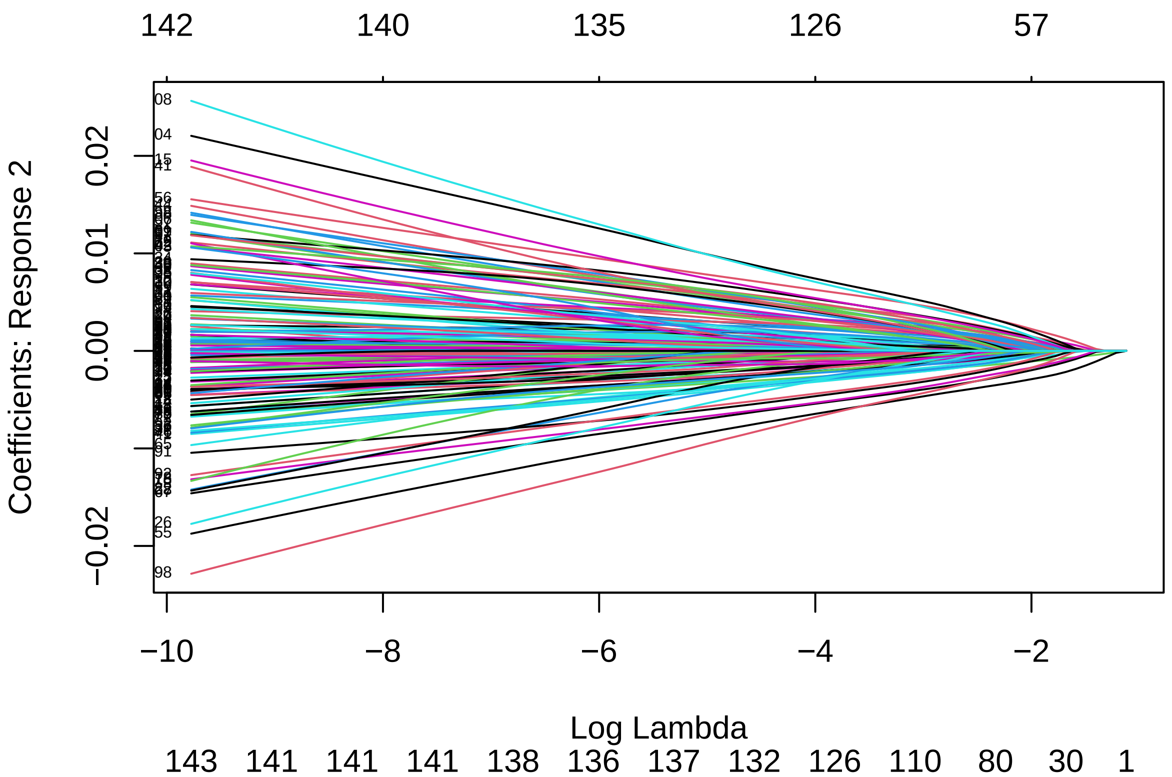


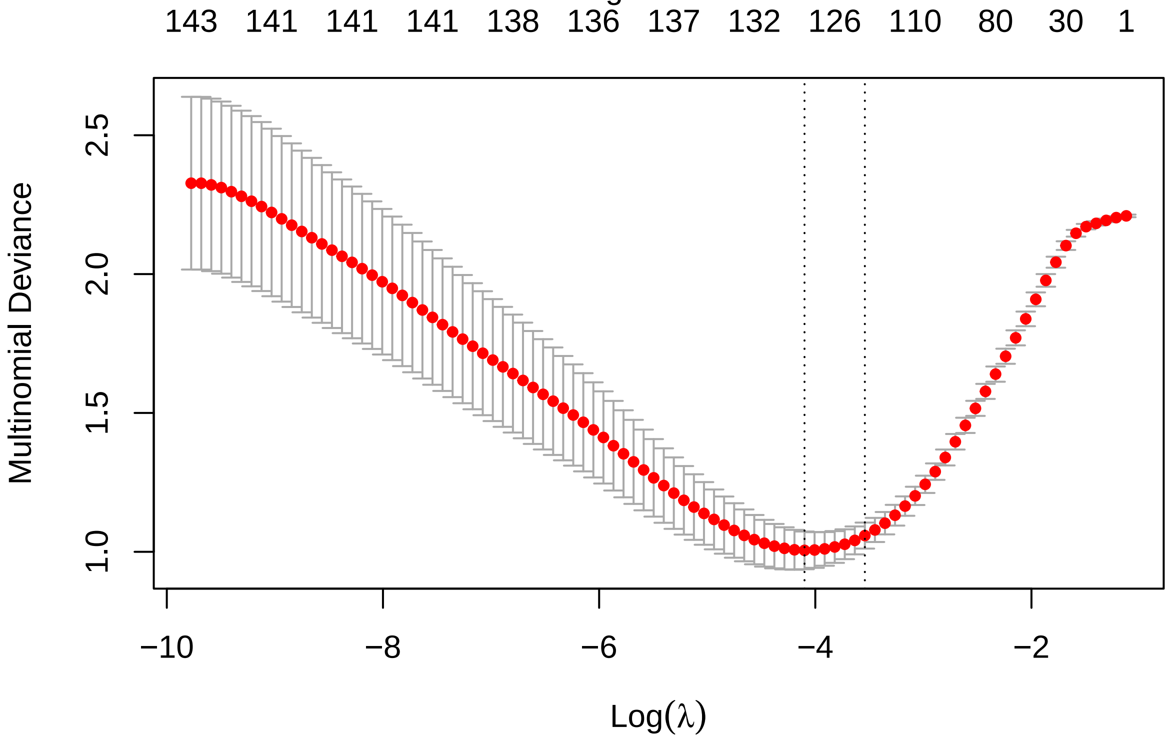


```
gm_mn <- mn_reg(pca_gm,weights=TRUE)
```

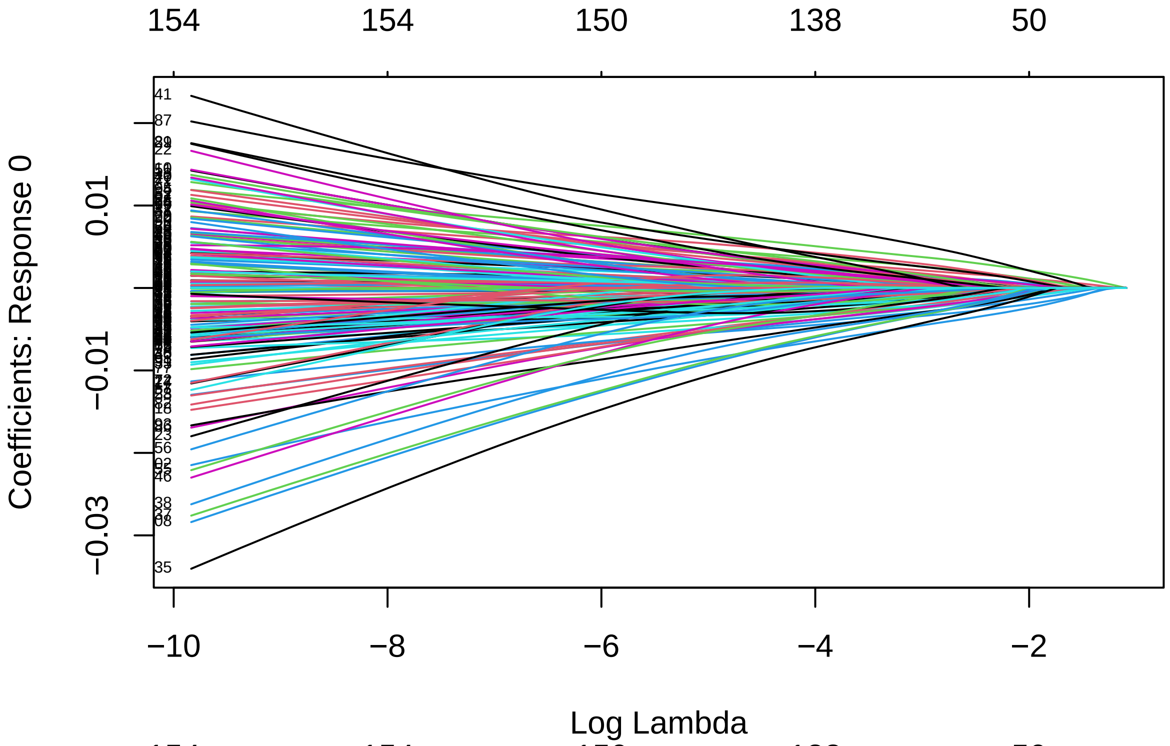

142
140
135
126
57
Coefficients: Response 0
Log Lambda

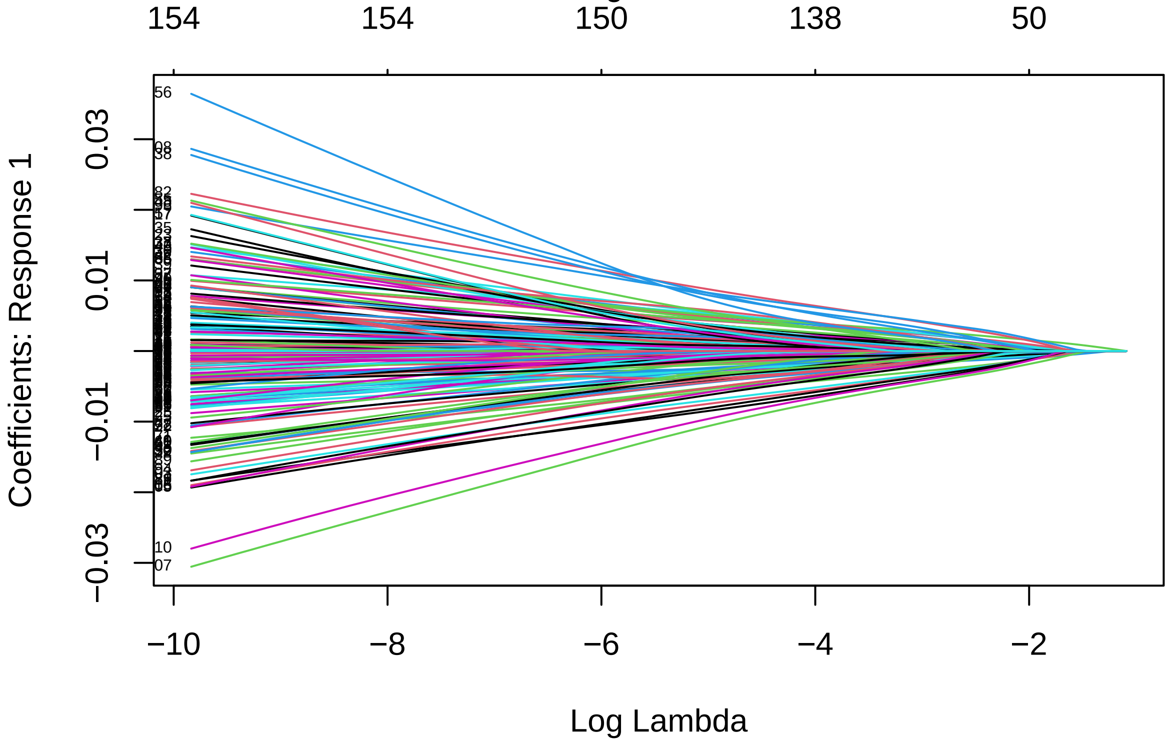

142
140
135
126
57
Coefficients: Response 1
Log Lambda

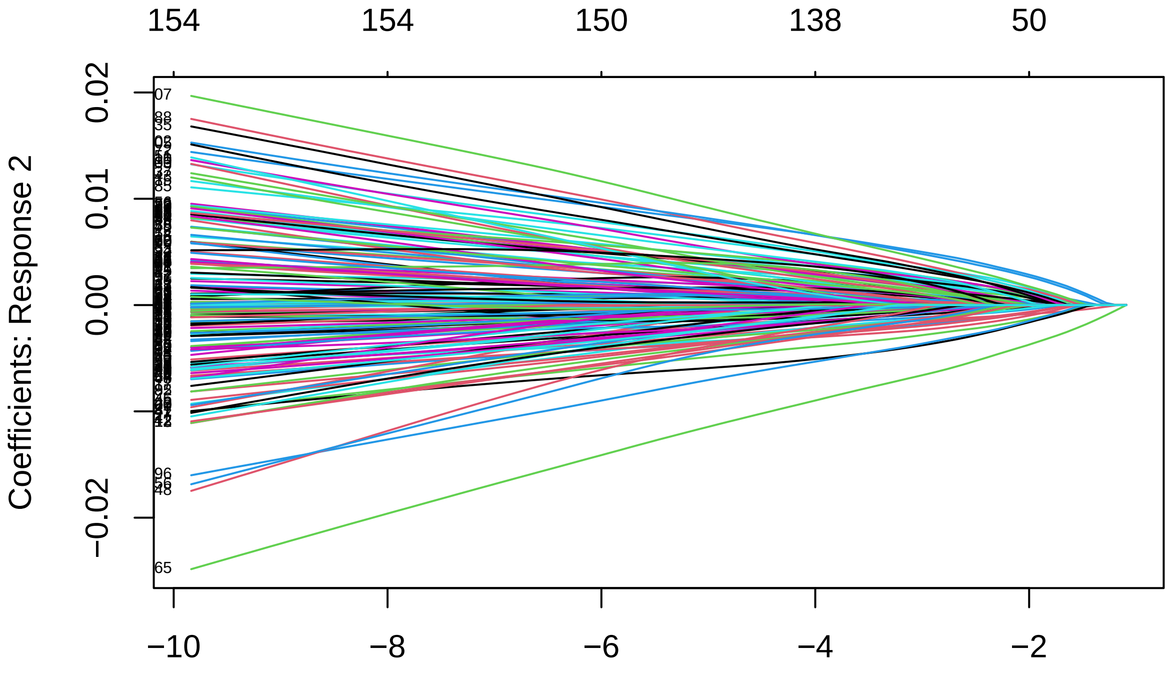


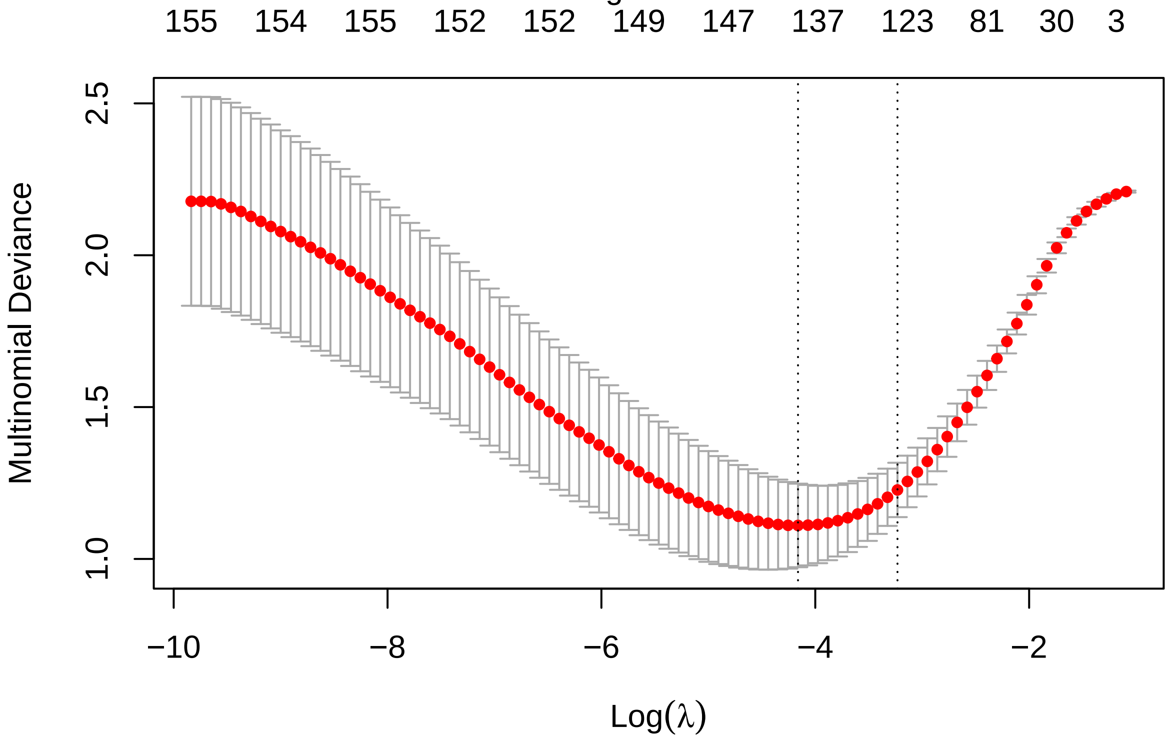


```
cb_mn <- mn_reg(pca_cb,weights=TRUE)
```

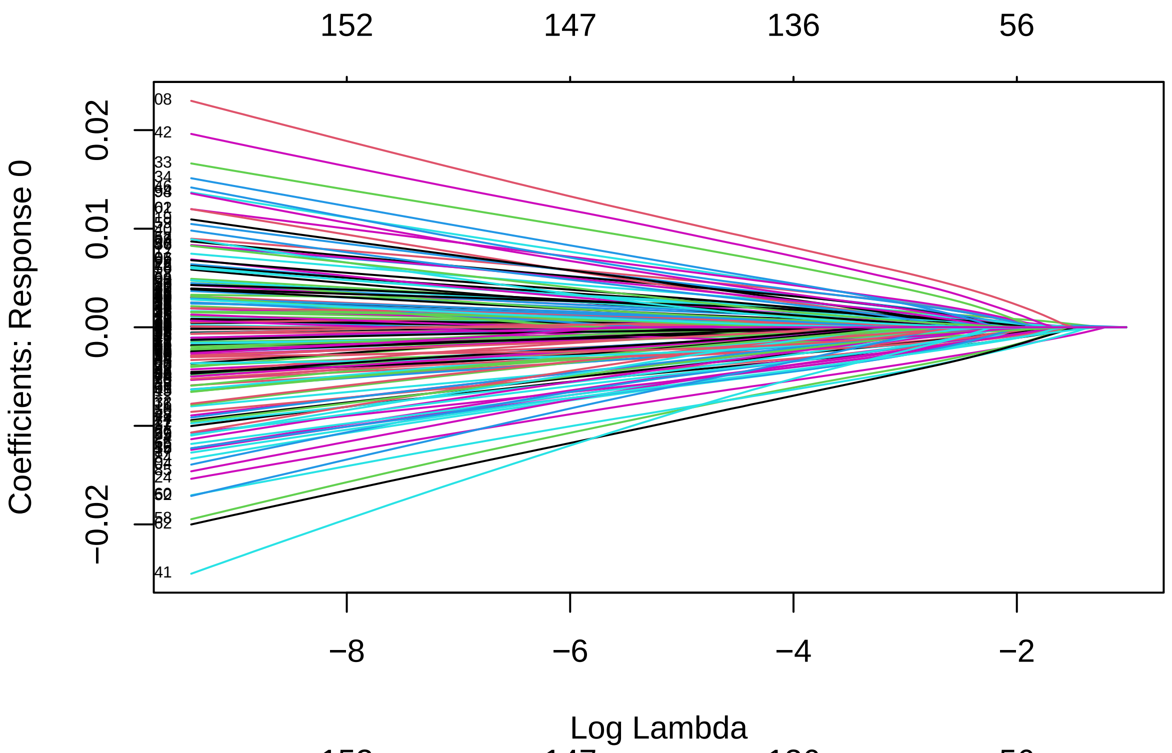

154
154
150
138
50
Coefficients: Response 0
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda


154
154
150
138
50
Coefficients: Response 1
0.03
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda

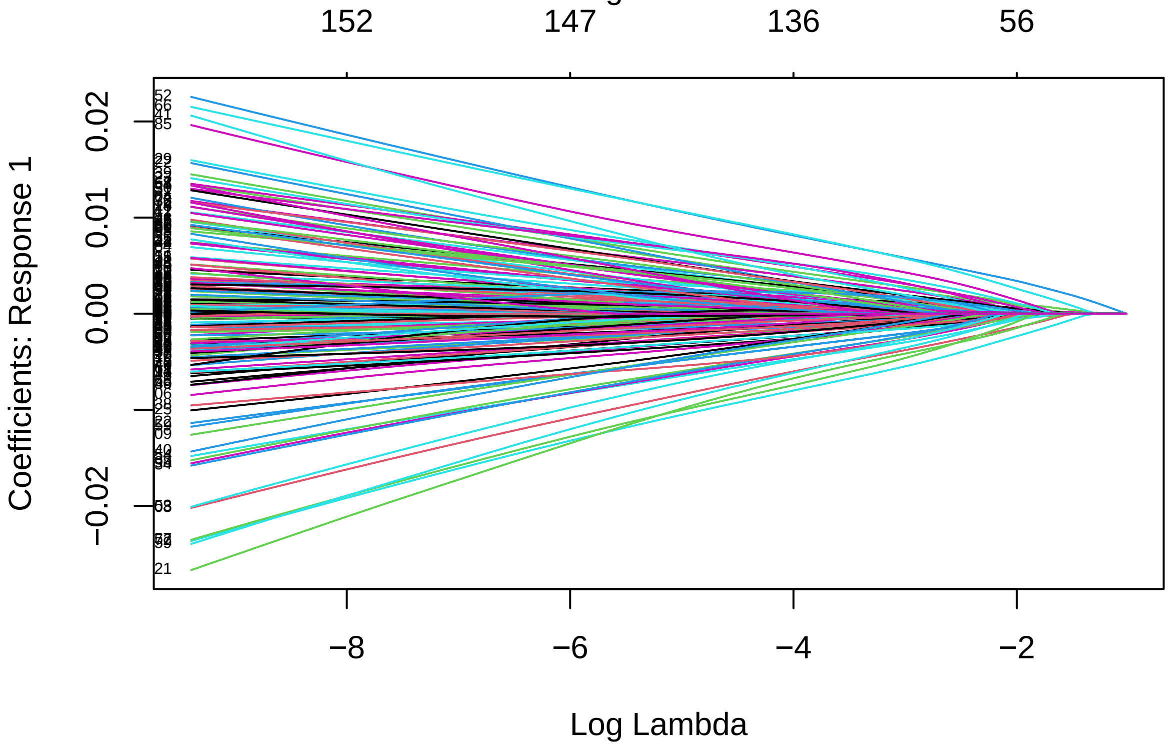




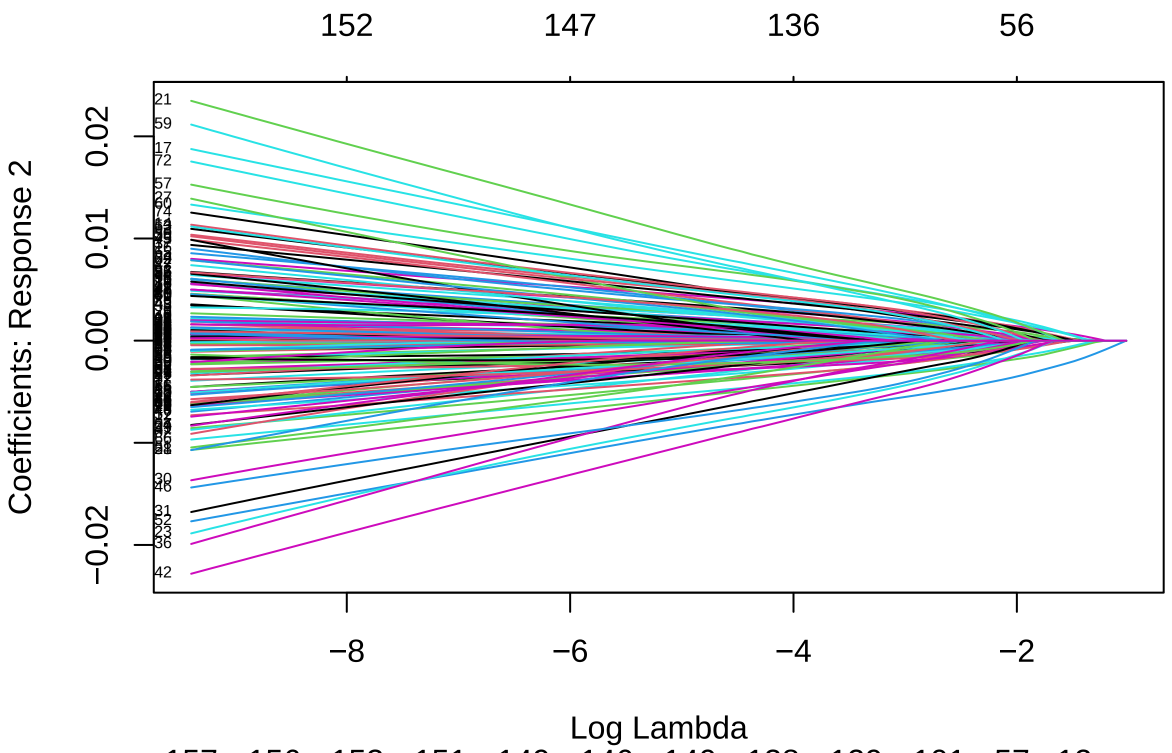
```
# balance weighted but no weights to accuracy penalty
wm_mn_npw <- mn_reg(pca_wm,weights=TRUE, w=FALSE)
```

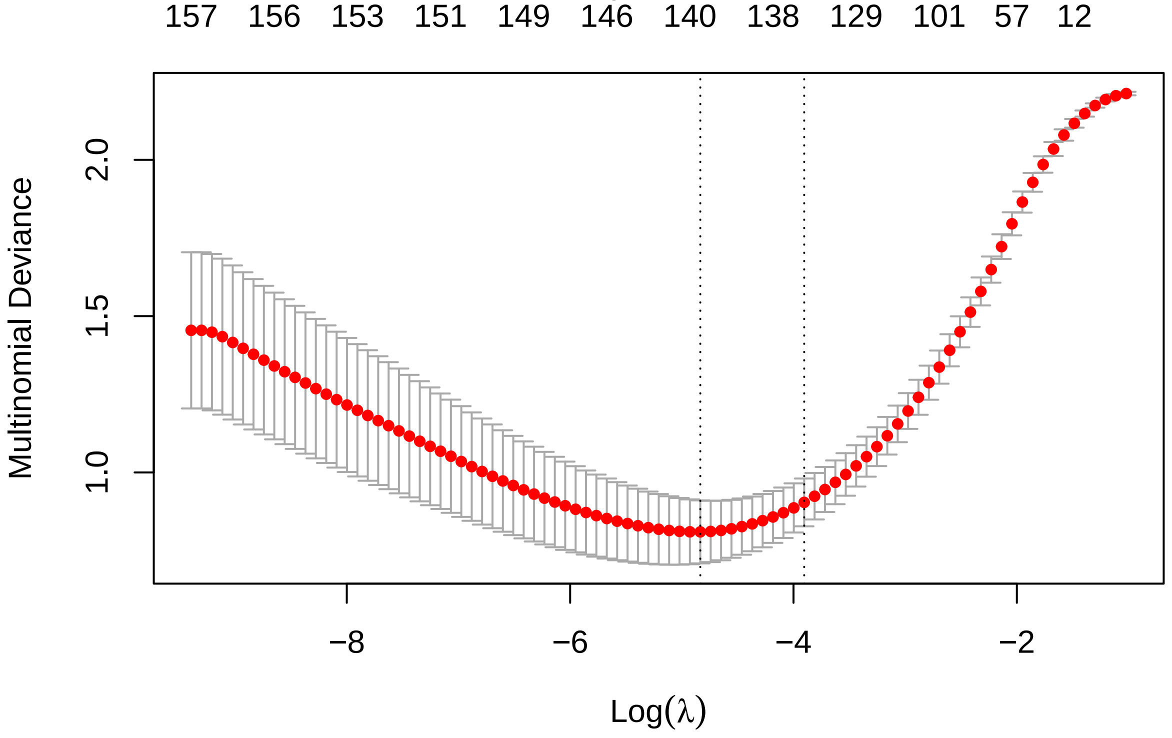


152
147
136
56
Coefficients: Response 0
0.02
0.01
0.00
−0.02
−8
−6
−4
−2
Log Lambda

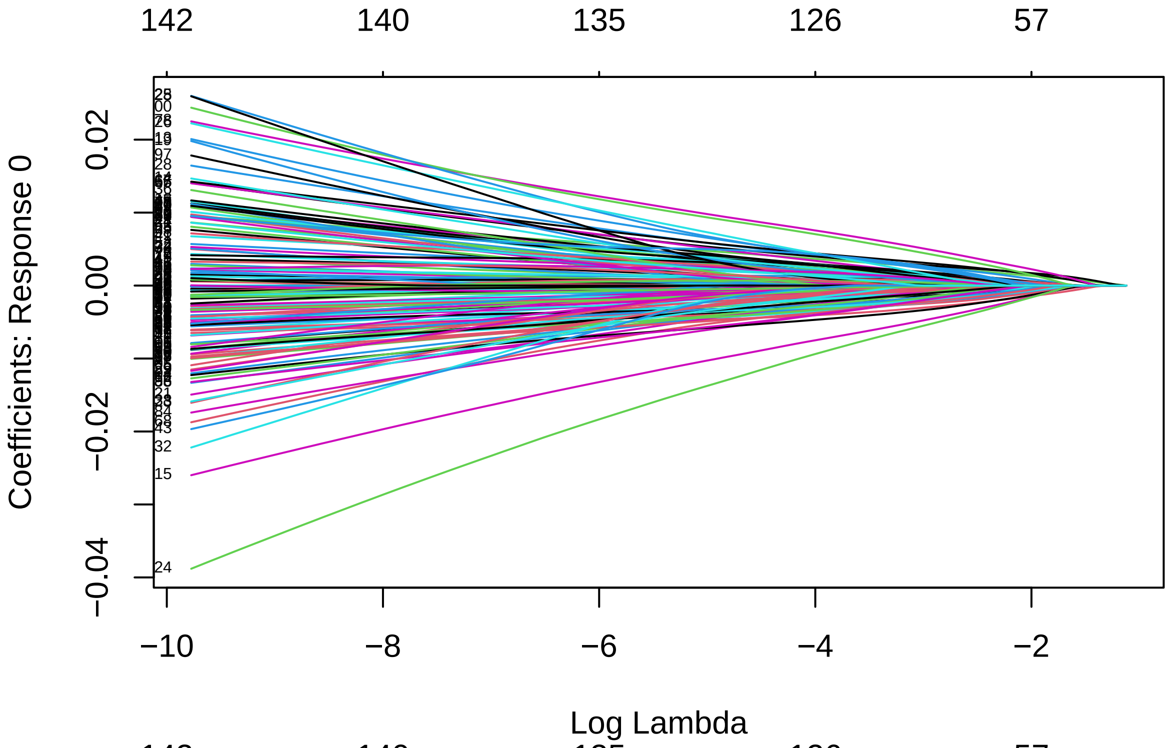


152
147
136
56
Coefficients: Response 1
0.02
0.01
0.00
−0.02
−8
−6
−4
−2
Log Lambda

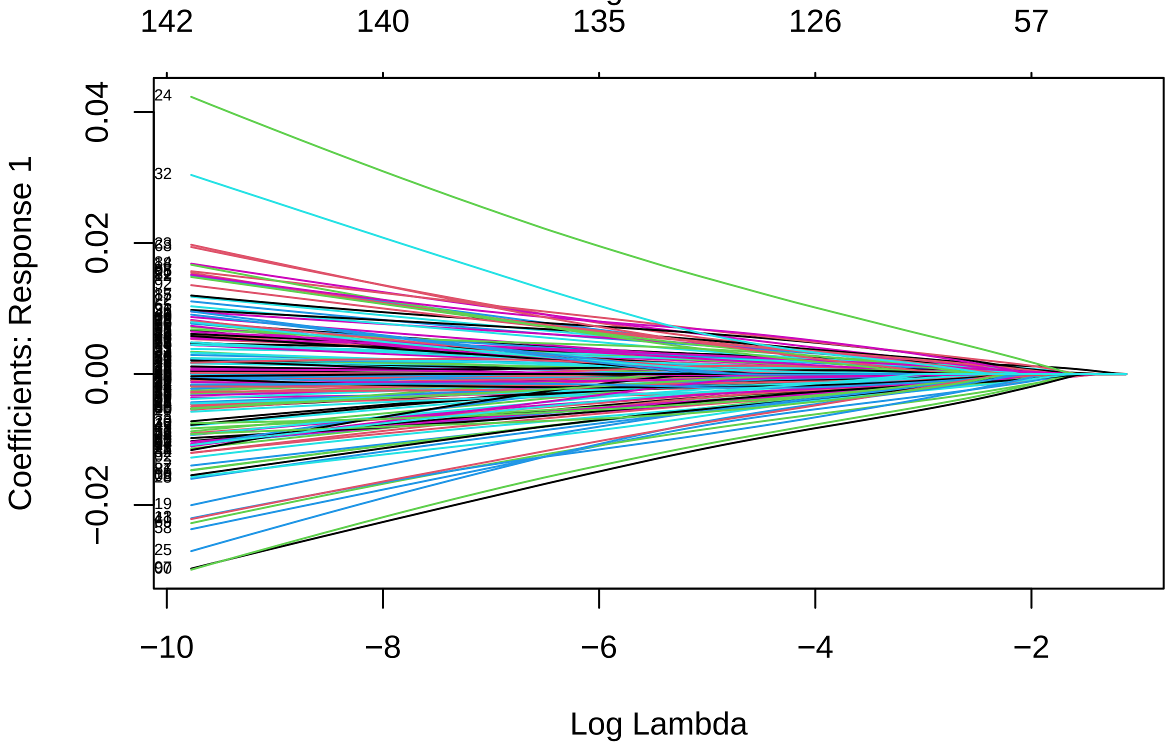




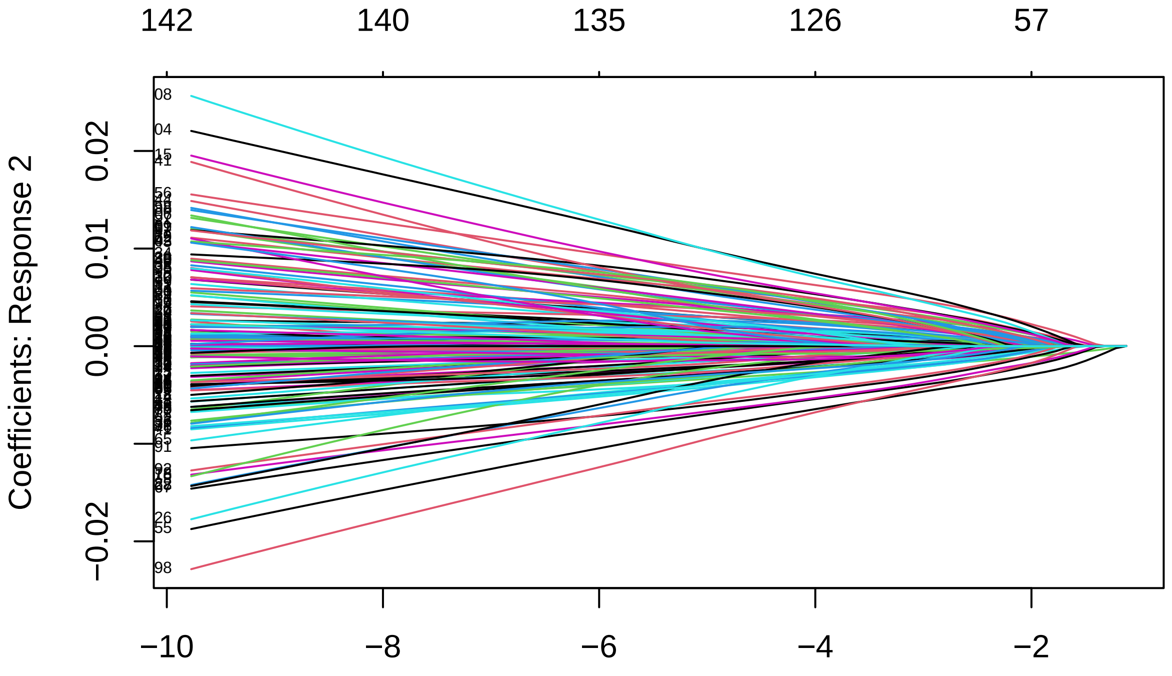
```
gm_mn_npw <- mn_reg(pca_gm,weights=TRUE, w=FALSE)
```

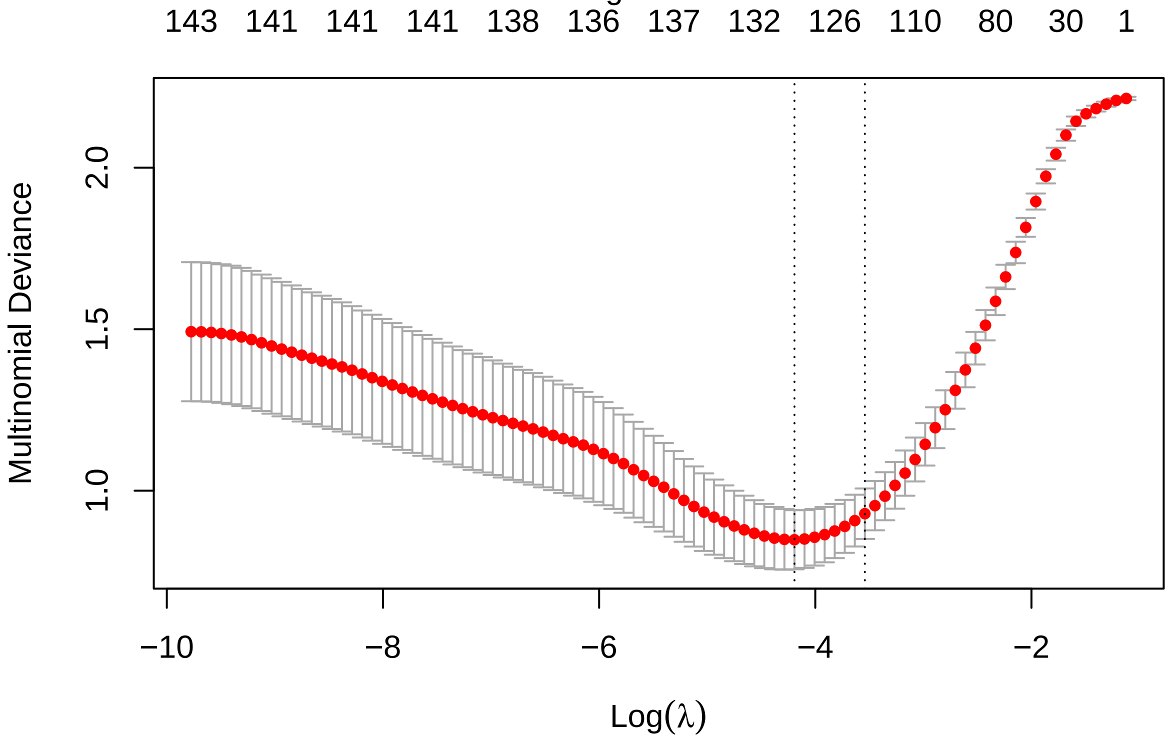


142
140
135
126
57
Coefficients: Response 0
0.02
0.00
−0.02
−0.04
−10
−8
−6
−4
−2
Log Lambda

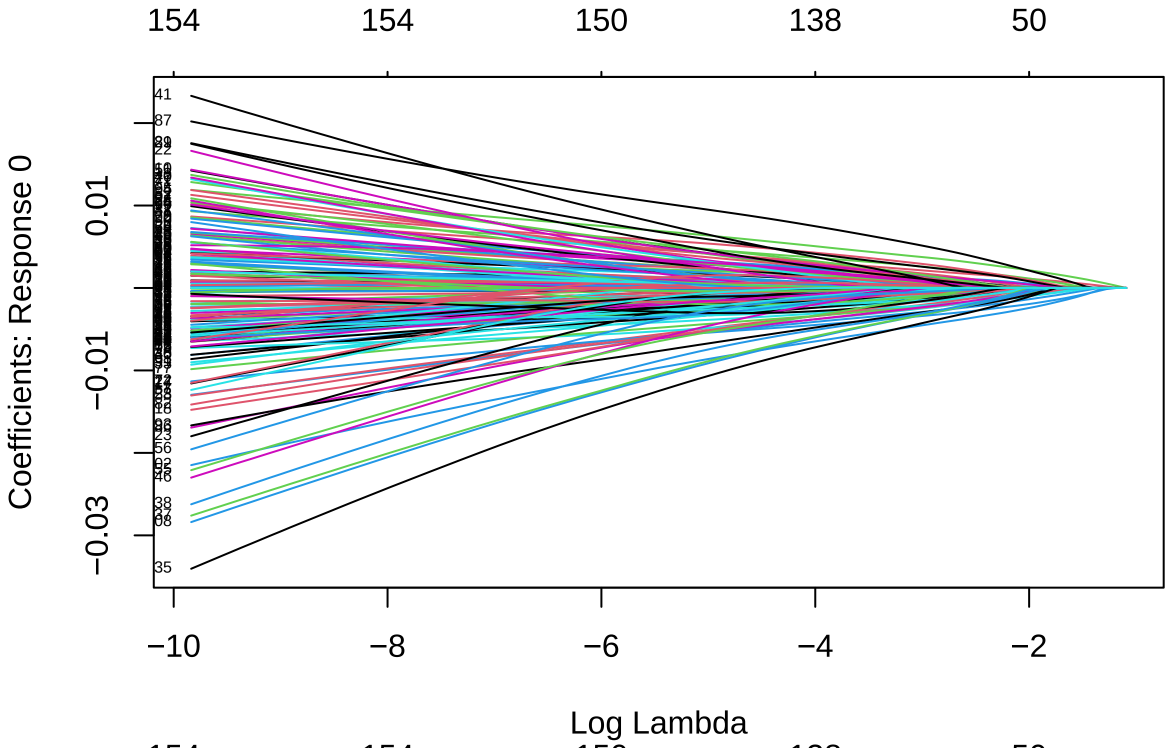


142
140
135
126
57
Coefficients: Response 1
0.04
0.02
0.00
−0.02
−10
−8
−6
−4
−2
Log Lambda

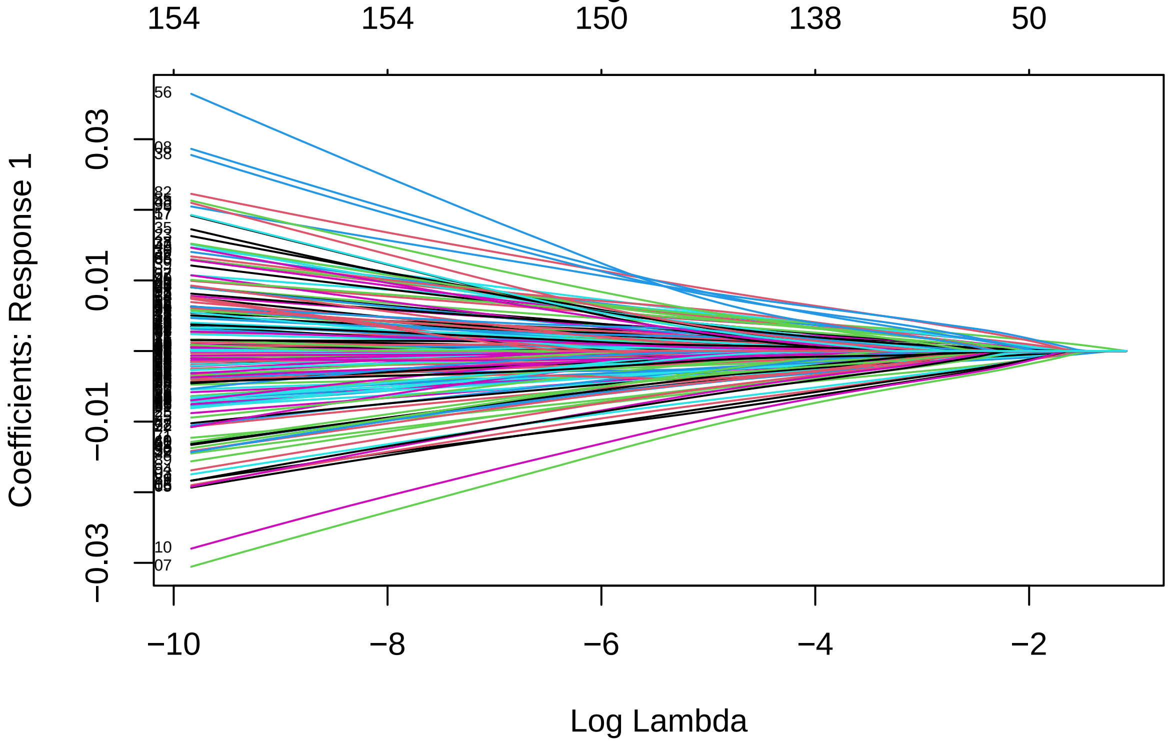




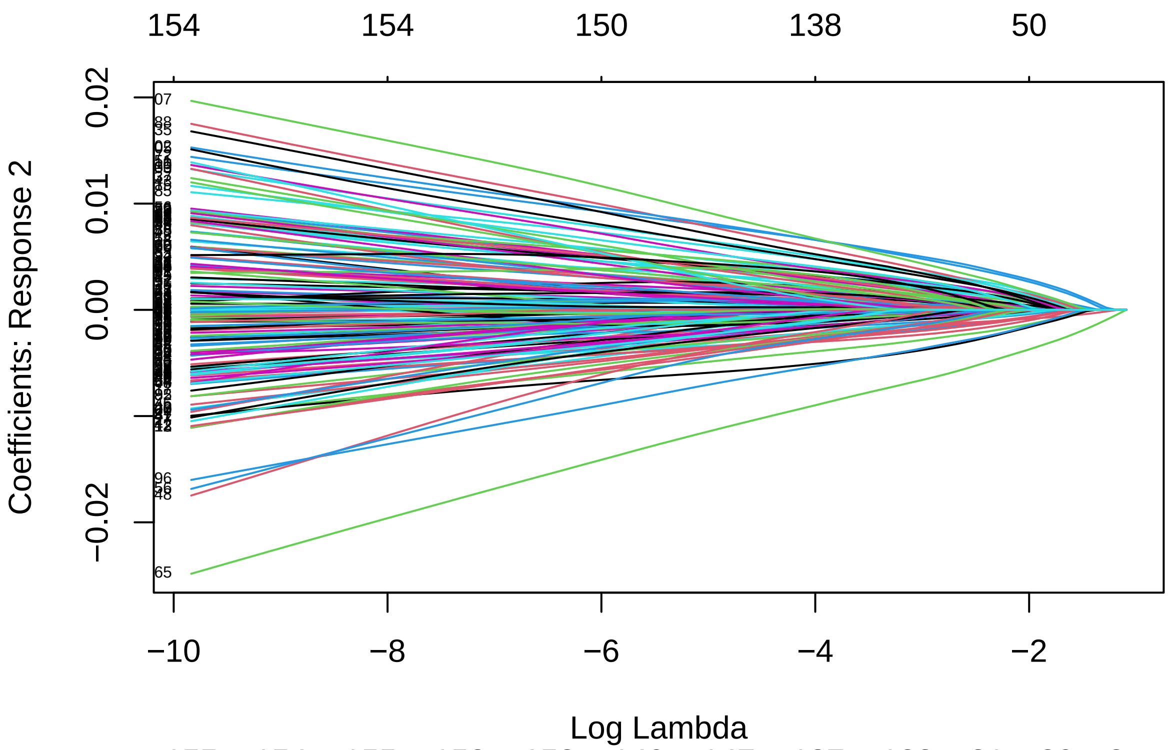
```
cb_mn_npw <- mn_reg(pca_cb,weights=TRUE, w=FALSE)
```

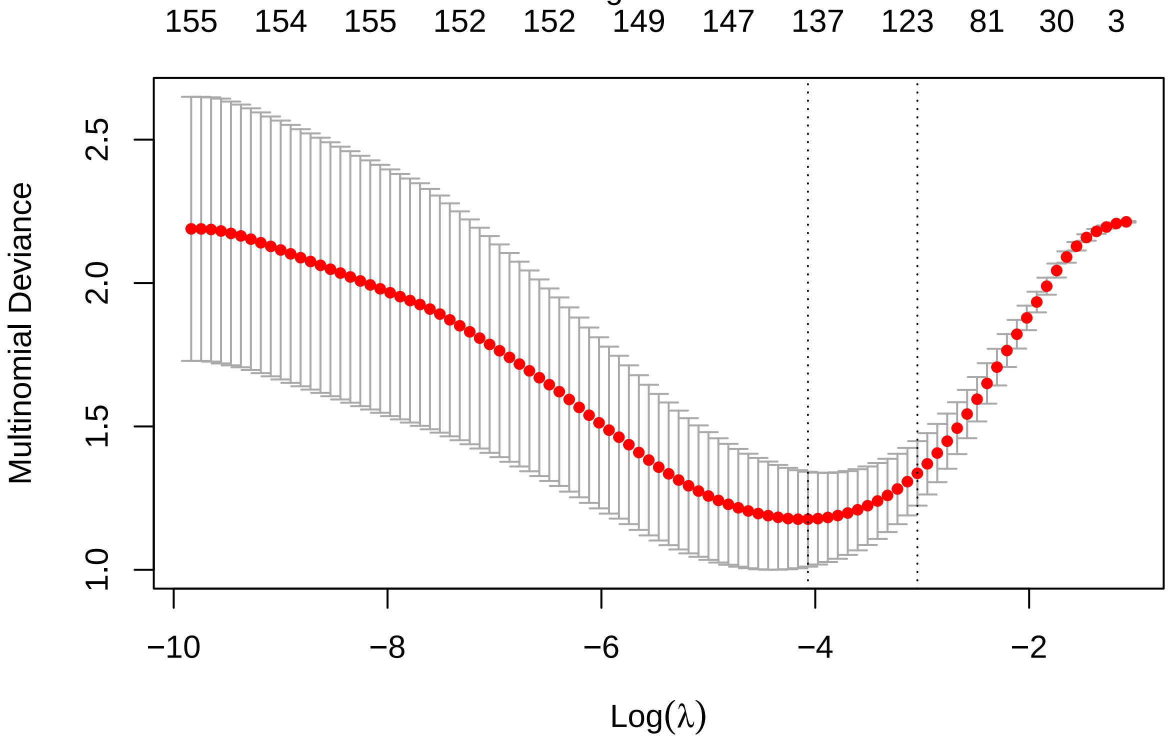


154
154
150
138
50
Coefficients: Response 0
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda



154
154
150
138
50
Coefficients: Response 1
0.03
0.01
−0.01
−0.03
−10
−8
−6
−4
−2
Log Lambda





#### 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  4  1
##          1  2 36  3
##          2  1  1 12
```

```
gm_mn$cm
```

```
##           Actual
## Prediction  0  1  2
##          0 29  4  1
##          1  0 37  2
##          2  1  0 13
```

```
cb_mn$cm
```

```
##           Actual
## Prediction  0  1  2
##          0 30  4  1
##          1  0 36  3
##          2  0  1 12
```

```
wm_mn_nw$cm
```

```
##           Actual
## Prediction  0  1  2
##          0 27  4  1
##          1  2 37  4
##          2  1  0 11
```

```
gm_mn_nw$cm
```

```
##           Actual
## Prediction  0  1  2
##           0 30  3  1
##           1  0 38  2
##           2  0  0 13
```

```
cb_mn_nw$cm
```

```
##            Actual
## Prediction  0  1  2
##           0 28  3  1
##           1  2 38  5
##           2  0  0 10
```

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

```
wm_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##           0 27  4  0
##           1  2 37  3
##           2  1  0 13
```

```
gm_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##           0 29  4  1
##           1  0 37  2
##           2  1  0 13
```

```
cb_lda$cm
```

```
##            Actual
## Prediction  0  1  2
##           0 29  3  1
##           1  0 37  3
##          2  1  1 12
```

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.

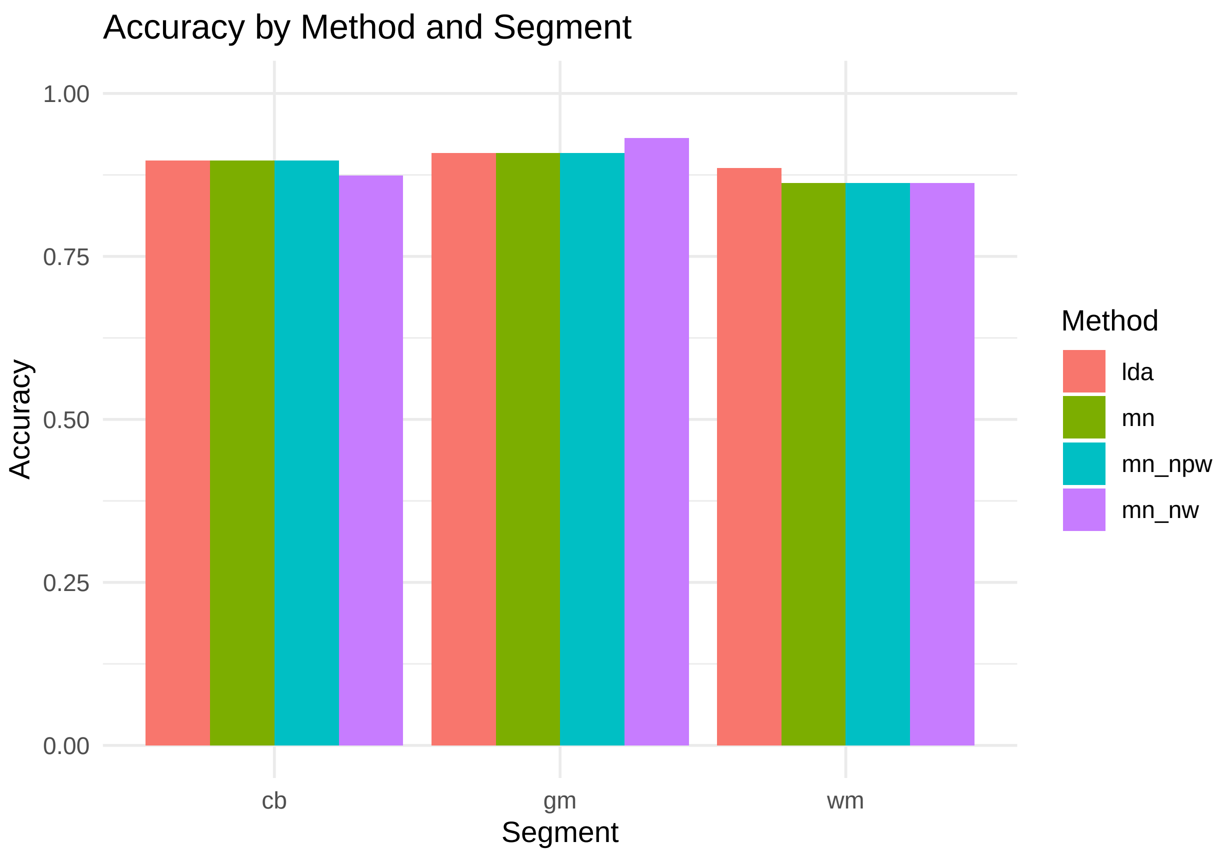
Plot method comparison

```
method <- c(rep("mn", 3),rep("mn_nw", 3), rep("mn_npw", 3), rep("lda", 3))
segment <- rep(c("wm", "gm", "cb"), 4)
accuracy <- c(wm_mn$accuracy, gm_mn$accuracy, cb_mn$accuracy,
              wm_mn_nw$accuracy, gm_mn_nw$accuracy, cb_mn_nw$accuracy,
              wm_mn_npw$accuracy, gm_mn_npw$accuracy, cb_mn_npw$accuracy,
              wm_lda$accuracy, gm_lda$accuracy, cb_lda$accuracy)

# combine the vectors into a data frame
```

```r
df <- data.frame(method = method, segment = segment, accuracy = accuracy)

kable(df, format = "markdown")
```

method	segment	accuracy
mn	wm	0.8620690
mn	gm	0.9080460
mn	cb	0.8965517
mn_nw	wm	0.8620690
mn_nw	gm	0.9310345
mn_nw	cb	0.8735632
mn_npw	wm	0.8620690
mn_npw	gm	0.9080460
mn_npw	cb	0.8965517
lda	wm	0.8850575
lda	gm	0.9080460
lda	cb	0.8965517

```r
ggplot(df, aes(x = segment, y = accuracy, fill = method)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Accuracy by Method and Segment",
       x = "Segment", y = "Accuracy",
       fill = "Method") +
  ylim(0, 1) +
  theme_minimal()
```



True Group Percentage

```r
df <- data.frame(wm_mn = calc_indiv_acc(wm_mn),
           gm_mn = calc_indiv_acc(gm_mn),
           cb_mn = calc_indiv_acc(cb_mn),
           wm_mn_nw = calc_indiv_acc(wm_mn_nw),
           gm_mn_nw = calc_indiv_acc(gm_mn_nw),
           cb_mn_nw = calc_indiv_acc(cb_mn_nw),
           wm_mn_npw = calc_indiv_acc(wm_mn_npw),
           gm_mn_npw = calc_indiv_acc(gm_mn_npw),
           cb_mn_npw = calc_indiv_acc(cb_mn_npw),
           wm_lda = calc_indiv_acc(wm_lda),
           gm_lda = calc_indiv_acc(gm_lda),
           cb_lda = calc_indiv_acc(cb_lda)
          )
rownames(df) <- c("CN","MCI","AD")
kable(t(df), caption='Group Accuracy % Per Model & Segment')
```

Table 2: Group Accuracy % Per Model & Segment

	CN	MCI	AD
wm_mn	0.844	0.878	0.857
gm_mn	0.853	0.949	0.929
cb_mn	0.857	0.923	0.923
wm_mn_nw	0.844	0.860	0.917
gm_mn_nw	0.882	0.950	1.000
cb_mn_nw	0.875	0.844	1.000
wm_mn_npw	0.844	0.878	0.857
gm_mn_npw	0.853	0.949	0.929
cb_mn_npw	0.857	0.923	0.923
wm_lda	0.871	0.881	0.929
gm_lda	0.853	0.949	0.929
cb_lda	0.879	0.925	0.857