Solutions Session 3

Marjolein Fokkema

```
student_full <- read.csv2("student-mat.csv")
```

1. Best subset, forward and backward selection

```
library("leaps")

## Best subset
model_BSS <- regsubsets(G3 ~ . , data = student_full, nvmax = 12,
                        method = "exhaustive")
sum <- summary(model_BSS)
which.max(sum$adjr2); which.max(sum$cp); which.min(sum$bic)
```

```
## [1] 12
```

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

```
## [1] 5
```

```
coef(model_BSS, id = 5)
```

```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

```
## Forward stepwise
model_Fstep <- regsubsets(G3 ~ . , data = student_full,
                        nvmax = 33, method = "forward")
sum <- summary(model_Fstep)
which.max(sum$adjr2); which.max(sum$cp); which.min(sum$bic)
```

```
## [1] 13
```

```
## [1] 33
```

```
## [1] 5
```

```
coef(model_Fstep, id = 5)
```

```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

```
## Backward stepwise
model_Bstep <- regsubsets(G3 ~ . , data = student_full,
                          nvmax = 33, method = "backward")
sum <- summary(model_Bstep)
which.max(sum$adjr2); which.max(sum$cp); which.min(sum$bic)
```

```
## [1] 13
```

```
## [1] 33
```

```
## [1] 5
```

```
coefs <- coef(model_Bstep, id = 5)
coefs
```

```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

With the BIC criterion, all three subset selection methods (best subset, forward stepwise, backward stepwise) selected the same variables, and thus yielded identical final models.

The strongest predictor of math achievement at moment 3 seems to be math achievement at moment 2 (which is not very surprising).

To generate predictions for the test observations, I created a design matrix with intercept and the selected predictor variables, then multiplied those by the coefficients I extracted from one of the models:

```
x_test <- as.matrix(cbind(1, test_dat[c("age", "famrel", "absences", "G1", "G2")]))
l0_preds <- x_test %*% coefs
```

Alternatively, you could also refit an OLS model to the selected predictors using function `lm()`, and then use the `predict` method to obtain predictions for the new observations.

Next, I computed mean squared error (MSE) for the test observations:

```
MSE_l0 <- mean((l0_preds - test_dat$G3)^2)
MSE_l0
```

```
## [1] 4.429321
```

```
MSE_max <- var(test_dat$G3) ## serves as a benchmark
MSE_max
```

```
## [1] 25.65129
```

```
1 - MSE_l0 / MSE_max ## cross-validated R2
```

```
## [1] 0.8273256
```

2. Ridge, Lasso, Elastic Net

I fitted a Lasso, Ridge and Elastic Net (with $\alpha = 0.5$) to the training data:

```
library("glmnet")
par(mfrow = c(1, 3))

## Data preparation
x <- model.matrix(G3 ~ . -1, train_dat)
x_test <- model.matrix(G3 ~ . -1, test_dat)
y <- train_dat$G3

## Model fitting
set.seed(42)
l_mod <- cv.glmnet(x, y, alpha = 1) ## l is for lasso
l_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.1835    34   3.458 0.617       8
## 1se 0.7409    19   3.997 0.717       1
```

```
plot(l_mod)
l_coefs <- coef(l_mod, s = "lambda.min") ## by default 1se criterion would be used
l_coefs[l_coefs[ , 1] != 0, ]
```

```
##    (Intercept)            age   Fjobservices guardianmother      higheryes
##   -1.375061189   -0.012579948   -0.088060234    0.075355766    0.080350572
##         famrel       absences             G1             G2
##    0.151786387    0.004628783    0.069702792    0.986738604
```

```
l_preds <- predict(l_mod, s = "lambda.min", newx = x_test)

set.seed(42)
r_mod <- cv.glmnet(x, y, alpha = 0) ## r is for ridge
r_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.3954   100   3.926 0.4304      42
## 1se 1.2075    88   4.297 0.4998      42
```

```r
plot(r_mod)
r_coefs <- coef(r_mod, s = "lambda.min")
r_coefs[r_coefs[ , 1] != 0, ]
```

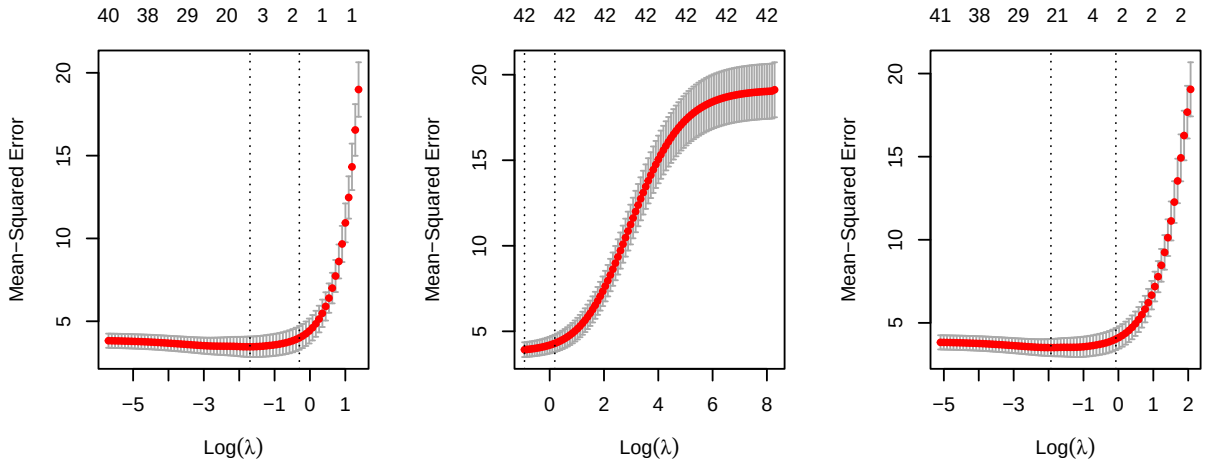
```
##      (Intercept)         schoolGP         schoolMS             sexM
##       0.42031180       0.01113638      -0.01157787      -0.10333452
##              age        addressU       famsizeLE3        PstatusT
##      -0.20542834       0.06925667       0.08403962       0.15805816
##             Medu             Fedu       Mjobhealth        Mjobother
##       0.13644781      -0.21655081      -0.11134998      -0.02274362
##     Mjobservices      Mjobteacher       Fjobhealth        Fjobother
##       0.38064590      -0.02537189       0.20957032       0.20249170
##     Fjobservices      Fjobteacher       reasonhome      reasonother
##      -0.21169932       0.01734250      -0.18229930       0.46842004
## reasonreputation   guardianmother    guardianother       traveltime
##       0.02980125       0.18329287      -0.08599172      -0.01586917
##        studytime         failures     schoolsupyes        famsupyes
##      -0.09902035      -0.21591193       0.26024621       0.08384962
##          paidyes    activitiesyes       nurseryyes        higheryes
##       0.27731464      -0.25437907      -0.32345171       0.58644628
##      internetyes      romanticyes           famrel         freetime
##      -0.41162797      -0.30524261       0.31294205      -0.03843056
##            goout             Dalc             Walc           health
##       0.13462310      -0.03948098       0.09939961       0.04759205
##         absences               G1               G2
##       0.03325087       0.30235559       0.76872311
```

```r
r_preds <- predict(r_mod, s = "lambda.min", newx = x_test)

set.seed(42)
e_mod <- cv.glmnet(x, y, alpha = .5) ## e is for elastic net
e_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 0.5)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.1448    44   3.514 0.5132      22
## 1se 0.9307    24   4.025 0.6617       2
```

```r
plot(e_mod)
```




```
e_coefs <- coef(e_mod, s = "lambda.min")
e_coefs[e_coefs[ , 1] != 0, ]
```

```
##    (Intercept)             age            Fedu   Mjobservices      Fjobother
##   -0.442981555   -0.129821244   -0.050720794    0.198921576    0.057010824
##   Fjobservices     reasonhome    reasonother guardianmother       failures
##   -0.254005929   -0.079120372    0.109566830    0.184854298   -0.097978405
##        paidyes activitiesyes     nurseryyes      higheryes    internetyes
##    0.094565940   -0.127442001   -0.113780592    0.404186683   -0.256272679
##    romanticyes         famrel          goout           Walc         health
##   -0.166071614    0.279118370    0.057398717    0.042904388    0.002637022
##       absences             G1             G2
##    0.024451458    0.149200666    0.930034024
```

```
e_preds <- predict(e_mod, s = "lambda.min", newx = x_test)
```

How many variables were selected depended on the λ criterion used for selecting the final model: The value that yielded the lowest cross-validated MSE (`lambda.min`) or the value that is higher but yielded MSE within 1 SE of the minimum (`lambda.1se`).

With `lambda.min`, 8, 22 and 42 variables were retained with Lasso, Elastic Net and Ridge, respectively. (Note that `model.matrix` coded all factors as sets of dummy variables, thus increasing the number of predictor variables to 42 instead of 33).

With `lambda.1se`, 1, 2 and 42 variables were retained with Lasso, Elastic Net and Ridge, respectively.

Finally, I computed test MSE:

```
MSE_l <- mean((l_preds - test_dat$G3)^2)
MSE_l
```

```
## [1] 4.955671
```

```
MSE_r <- mean((r_preds - test_dat$G3)^2)
MSE_r
```

```
## [1] 5.525793
```

```
MSE_e <- mean((e_preds - test_dat$G3)^2)
MSE_e
```

```
## [1] 4.93186
```

```
1 - MSE_l / MSE_max ## cross-validated R2
```

```
## [1] 0.8068062
```

```
1 - MSE_r / MSE_max ## cross-validated R2
```

```
## [1] 0.7845803
```

```
1 - MSE_e / MSE_max ## cross-validated R2
```

```
## [1] 0.8077344
```

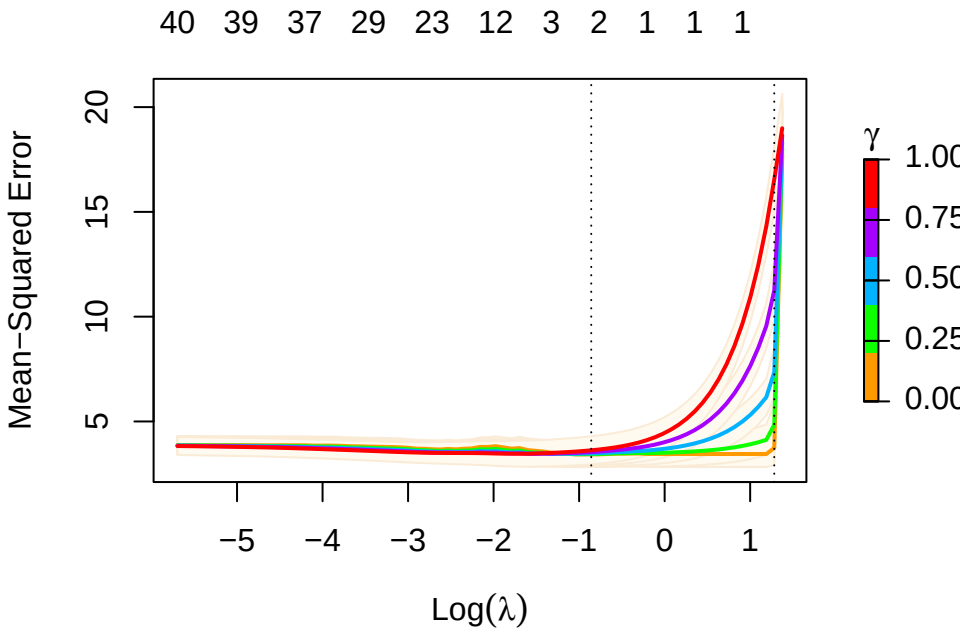
Among the shrinkage methods, elastic net performed best. But all three shrinkage methods were outperformed by the subset selection methods.

3. Relaxed lasso

```
set.seed(42)
rl_mod <- cv.glmnet(x, y, relax = TRUE)
rl_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, relax = TRUE)
##
## Measure: Mean-Squared Error
##
##     Gamma Index Lambda Index Measure     SE Nonzero
## min     0     1  0.424    25   3.429 0.6036       2
## 1se     0     1  3.603     2   3.736 0.8216       1
```

```
plot(rl_mod)
```



The relaxed Lasso retained only 1 or 2 predictors, depending on the criterion used. Both λ values had an optimal γ value of 0, indicating that variables are selected using the Lasso penalty, but no shrinkage should be performed when estimating the coefficients.

```
rl_coefs <- coef(rl_mod, s = "lambda.min")
rl_coefs[rl_coefs[ , 1] != 0, ]
```

```
## (Intercept)          G1          G2
##   -1.510367    0.105165    1.010467
```

```
rl_coefs <- coef(rl_mod, s = "lambda.1se")
rl_coefs[rl_coefs[ , 1] != 0, ]
```

```
## (Intercept)          G2
##   -1.247066    1.092904
```

The relaxed lasso retained the G2 variable as the strongest predictor, followed by the G1 variable. It appears that past math performance is the best predictor of future math performance.

```
rl_preds <- predict(rl_mod, s = "lambda.min", newx = x_test)
MSE_rl <- mean((rl_preds - test_dat$G3)^2)
MSE_rl
```

```
## [1] 4.86109
```

```
1 - MSE_rl / MSE_max ## cross-validated R2
```

```
## [1] 0.8104933
```

The relaxed Lasso outperformed the other three shrinkage methods on the test data. It was, however, outperformed by the subset selection methods. Note that the relaxed Lasso could also be termed a selection (not shrinkage) method with $\gamma = 0$: It is then forward stepwise selection with variable entry determined by the Lasso.

I would conclude that relaxed Lasso shows a very good accuracy-complexity trade-off, as it uses only 2 predictors and yields performance on the test set that is very close to the winning subset selection models that used 5 variables for prediction.

4. Predicting Depressive Disorder using Questionnaire Items

Read in data:

```
train <- readRDS("masq_train.Rda")
test <- readRDS("masq_test.Rda")
```

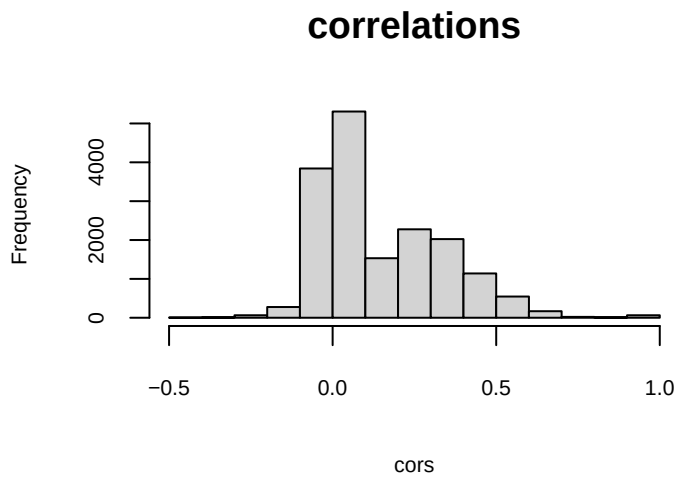
Prepare predictors and response in training and test set for analyses with `cv.glmnet`:

```
library("glmnet")
x <- model.matrix(D_DEPDYS ~ ., data = train)
y <- train$D_DEPDYS
x_test <- model.matrix(D_DEPDYS ~ ., data = test)
y_test <- test$D_DEPDYS
```

To choose the regularization method, we should think about the properties of the data, and think about how the results might be interpreted and applied.

To choose between ridge and lasso methods, the correlation between predictors is relevant:

```
cors <- cor(x[,-1])
diag(cors) <- NA
hist(cors, main = "correlations", cex.lab = .7, cex.axis=.7)
```



There are substantial correlations between predictors, thus some ridge penalization will likely be beneficial for prediction.

For interpretation and application, a sparse model with only few predictors would likely be useful. For example, if only a subset of items is relevant for classifying depressed versus not, we could administer only a subset of items to future patients, reducing assessment burden. Some lasso penalization would thus be useful.

The relaxed lasso could further reduce the number of predictors retained, without damaging predictive accuracy too much, because it eases shrinkage on large coefficients with through $\gamma < 1$.

There is no single correct choice, important is to make an informed choice. I opted to fit a relaxed lasso (which also includes the standard lasso), a ridge model and an elastic net (which combines the ridge and lasso penalties) with α of .25:

```
## ridge
set.seed(42)
r_mod <- cv.glmnet(x = x, y = y, family = "binomial", alpha = 0)
r_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, family = "binomial", alpha = 0)
##
## Measure: Binomial Deviance
##
##     Lambda Index Measure      SE Nonzero
## min 0.1439    81   1.033 0.02579     132
## 1se 0.8428    62   1.057 0.01893     132
```

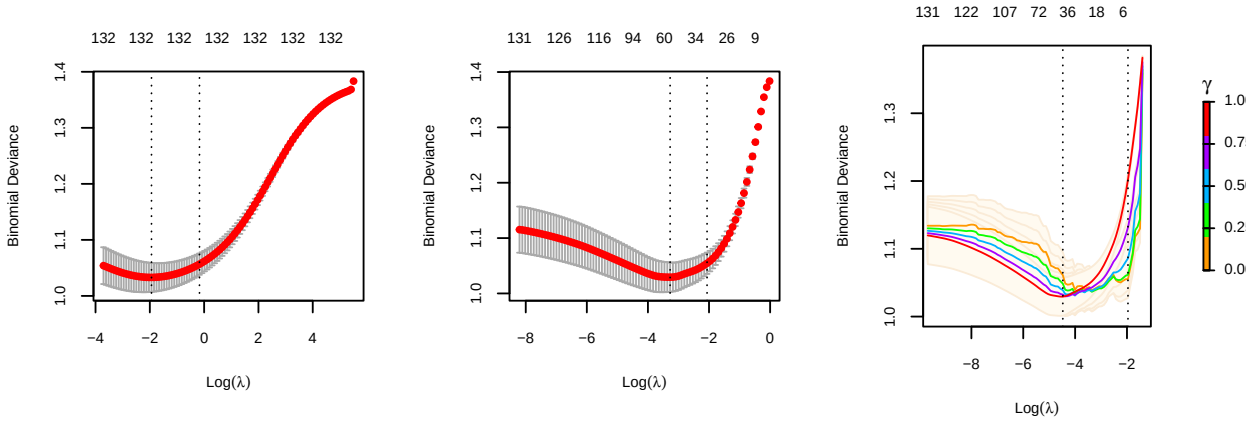
```
## elastic net alpha .25
set.seed(42)
en_mod <- cv.glmnet(x = x, y = y, family = "binomial", alpha = 0.25)
en_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, family = "binomial", alpha = 0.25)
##
## Measure: Binomial Deviance
##
##      Lambda Index Measure      SE Nonzero
## min 0.03787    36   1.029 0.02650      54
## 1se 0.12692    23   1.054 0.01926      32
```

```
## relaxed lasso
set.seed(42)
rl_mod <- cv.glmnet(x = x, y = y, relax = TRUE, family = "binomial")
rl_mod
```

```
##
## Call:  cv.glmnet(x = x, y = y, relax = TRUE, family = "binomial")
##
## Measure: Binomial Deviance
##
##     Gamma Index Lambda Index Measure      SE Nonzero
## min     1     5 0.0114    34   1.029 0.02803      37
## 1se     0     1 0.1406     7   1.055 0.02820       4
```

```
par(mfrow = c(1, 3))
plot(r_mod); plot(en_mod); plot(rl_mod)
```



Lowest cross-validated deviance is 1.029, obtained for the elastic net and relaxed lasso, when using the `lambda.min` criterion. For the relaxed lasso, this yields a model with 37 predictors, and a γ of 1 should be employed (i.e., the original lasso fit!).

One may prefer the more conservative (and more sparse and stable) `lambda.1se` criterion (the default in `glmnet`). Then the minimum deviance is obtained with elastic net, and relaxed lasso is very close.

We evaluate the best-fitting model(s) on the test data:

```
y_test <- test$D_DEPDYS
```

```
#####################################################
##
## lambda.1se criterion (sparsity preferred)
##

## Elastic net (alpha .25)
en_preds <- predict(en_mod, newx = x_test)
1 - sum(diag(table(en_preds > 0, y_test)))/nrow(train) ## MCR
```

```
## [1] 0.2335929
```

```
en_probs <- 1 / (1 + exp(-en_preds))
mean((en_probs - y_test)^2) ## Brier
```

```
## [1] 0.167805
```

```
## Relaxed lasso
rl_preds <- predict(rl_mod, newx = x_test)
1 - sum(diag(table(rl_preds > 0, y_test)))/nrow(train) ## MCR
```

```
## [1] 0.2486096
```

```
rl_probs <- 1 / (1 + exp(-rl_preds))
mean((rl_probs - y_test)^2) ## Brier
```

```
## [1] 0.1666514
```

Relaxed lasso wins in terms of Brier score, elastic net wins in terms of MCR.

```
############################################################
##
## Lambda.min criterion (predictive accuracy preferred)
##

## Elastic net (alpha .25)
en_preds <- predict(en_mod, newx = x_test, s = "lambda.min")
1 - sum(diag(table(en_preds > 0, y_test)))/nrow(train) ## MCR
```

```
## [1] 0.2296997
```

```
en_probs <- 1 / (1 + exp(-en_preds))
mean((en_probs - y_test)^2) ## Brier
```

```
## [1] 0.1616273
```

```
## Relaxed lasso
rl_preds <- predict(rl_mod, newx = x_test, s = "lambda.min")
1 - sum(diag(table(rl_preds > 0, y_test)))/nrow(train) ## MCR
```

```
## [1] 0.2263626
```

```
rl_probs <- 1 / (1 + exp(-rl_preds))
mean((rl_probs - y_test)^2) ## Brier
```

```
## [1] 0.1613809
```

For `lambda.min`, relaxed lasso wins in terms of Brier score and MCR.

We inspect which items were selected for prediction:

```
## Custom function to get item counts per subscale
ss_counts <- function(x) {
  AD <- c(1, 14, 18, 21, 23, 26, 27, 30, 33, 35, 36, 39, 40, 44, 49, 53, 58,
          66, 72, 78, 86, 89)
  AA <- c(3, 19, 25, 45, 48, 52, 55, 57, 61, 67, 69, 73, 75, 79, 85, 87, 88)
  GDD <- c(6, 8, 10, 13, 16, 22, 24, 42, 47, 56, 64, 74)
  GDA <- c(2, 9, 12, 15, 20, 59, 63, 65, 77, 81, 82)
  GDM <- c(4, 5, 17, 29, 31, 34, 37, 50, 51, 70, 76, 80, 83, 84, 90)
  sel_items <- as.numeric(substr(names(x), start = 5, stop = 7))
  res <- c(sum(sel_items %in% AD), sum(sel_items %in% AA),
           sum(sel_items %in% GDD), sum(sel_items %in% GDA),
           sum(sel_items %in% GDM))
  names(res) <- c("AD", "AA", "GDD", "GDA", "GDM")
  res
}

################################################
##
```

```r
## lambda.1se criterion (sparsity preferred)
##

## elastic net (alpha .25)
en_coefs <- as.matrix(coef(en_mod))
ss_counts(en_coefs[en_coefs != 0, ][-1])
```

```
##  AD  AA GDD GDA GDM
##  12   0   5   0   8
```

```r
## relaxed lasso
rl_coefs <- as.matrix(coef(rl_mod))
ss_counts(rl_coefs[rl_coefs != 0, ][-1])
```

```
##  AD  AA GDD GDA GDM
##   2   0   2   0   0
```

```r
############################################################
##
## Lambda.min criterion (predictive accuracy preferred)
##

## relaxed lasso
rl_coefs <- as.matrix(coef(rl_mod, s = "lambda.min"))
ss_counts(rl_coefs[rl_coefs != 0, ][-1])
```

```
## Warning in ss_counts(rl_coefs[rl_coefs != 0, ][-1]): NAs introduced by coercion
```

```
##  AD  AA GDD GDA GDM
##   8   1   4   2   7
```

Most items tend to be selected from the AD (Anhedonic Depression scale). This makes a lot of sense, because the prediction target is depression. For other disorder types (e.g., anxiety), items from other subscales may be more informative.