1.

a)

The model using best subset selection would perform the best, although it is also possible that forward & backward stepwise selection could be used at the same model.

This is because best subset selection will fit all 2p possible models using the p predictors. If the selection criteria is minimizing the training RSS, there is no subset of predictors that could be identified by forward/backward selection that wouldn’t also be identified by best subset selection.

b)

The best subset selection is more likely to perform better. The selection process for all three algorithms will usually be based on a combination of minimizing the training RSS (to choose between models of the same size), followed by using cross-validated prediction error or some sort of penalized statistic (e.g. AIC, BIC, Adjusted R2) to select the final model from the p+1 candidate models. In the case of p = 20, best subset will select from 2p=220=1,048,576 models, whereas the others will select from ∑p−1k=0(p−k)=1+p(p+1)/2=1+20(20+1)/2=211 models.

c)

1. It is true since the (k+1)-variable model will be identical to the k-variable model, but with one additional predictor.
2. It is true since the k-variable model will be identical to the (k+1)-variable model, but with one predictor removed.
3. It is false, since forwards and backwards stepwise selection have different starting points (the null model and the full model) and will take different selection paths. The statement could hold true for specific examples, but it is not generally true.
4. It is false. The reasoning is familiar to iii).
5. It is false since there is no guarantee that the best variable subset of size (k+1) will simply be the best variable subset of size k with one additional predictor. If this were the case, we could simply do forward selection and reduce the number of models we test significantly.

2a)

I chose Part iii). This is because lasso selects the β^ that minimizes RSS+λ∑pi=1|βi|, instead of simply the RSS in least squares.

Since the shrinkage penalty λ∑pi=1|βi| is small for β1,β2,…,βp close to zero, this tends to shrink the estimates towards zero (because for a given λ>0, the optimal lasso β^ will be closer to zero than the least squares β^). For a larger λ, the shrinkage terms importance is higher relative to the RSS, so the shrinkage increases. This shrinkage is what reduces the variance of the predictions, at the cost of a small increase in bias.

4.

a) It will steadily increase. When λ = 0, the ridge regression β^ will be the same as the least squares estimate for β (since the shrinkage term is removed), which will already minimize the training RSS. As λ increases, this training RSS can only increase, and will do so as the shrinkage increases.

b) It will decrease initially, and then eventually start increasing in a U shape. As λ (shrinkage) increases, the hope is that the reduction in variance will outweigh the cost of shrinking the β^’s towards zero. This will generally mean that the test RSS will decrease, up to the point where increased shrinkage simply results in the model underfitting and decreasing its prediction accuracy (where the increased bias outweighs the decreased variance), at which point the test RSS will start increasing again.

c) It will steadily decrease. Increasing λ decreases the flexibility because the β’s are shrunk towards zero, reducing the variance. We can continue increasing the shrinkage arbitrarily, reducing the variance until it is arbitrarily close to zero (as the β^’s approach zero, the model approaches the null model and our predictions approach zero variance).

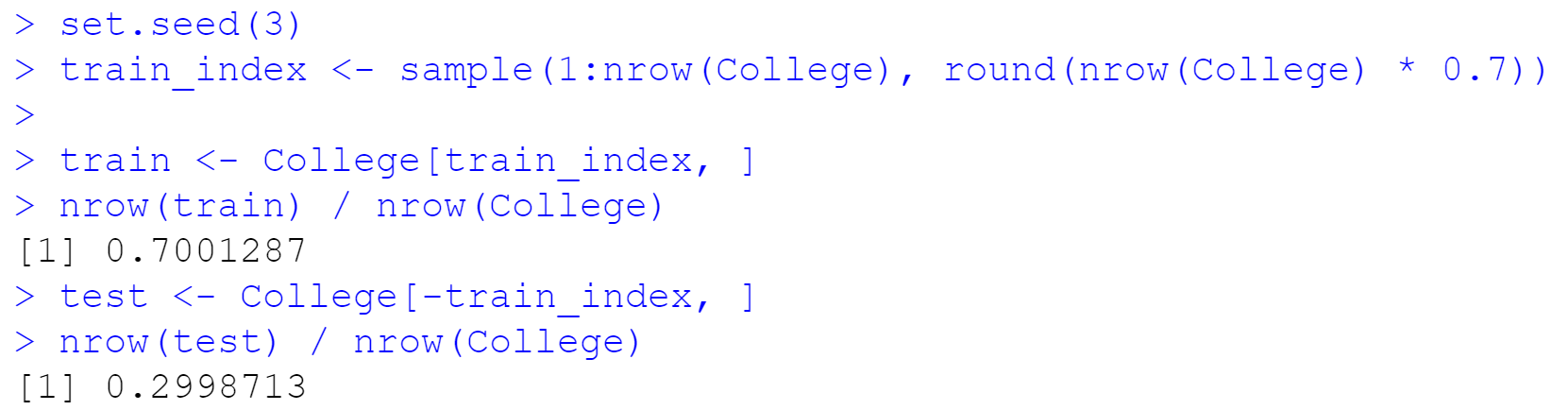
d) It will steadily increase. It is the same reasoning as part (c) - increasing λ (decreasing the flexibility) will increase the bias as the β’s are shrunk towards zero.

e) It will remain constant. It is the same reasoning as in question 3 - the irreducible error is model-independent.

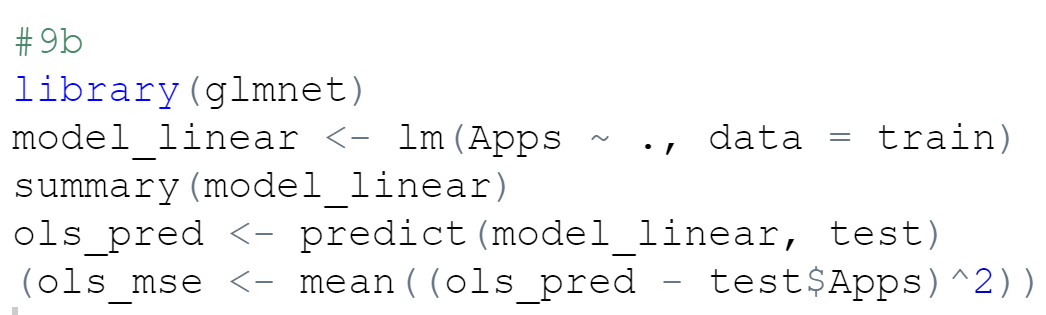
9.

a)

I randomly sample 70% of the observations for “train” set.



b)



2019409

The test error is 20219409.

c)

train\_mat <- dummyVars(Apps ~ ., data = train, fullRank = F) %>%

predict(newdata = train) %>%

as.matrix()

test\_mat <- dummyVars(Apps ~ ., data = test, fullRank = F) %>%

predict(newdata = test) %>%

as.matrix()

set.seed(3)

model\_ridge <- cv.glmnet(y = train$Apps,

x = train\_mat,

alpha = 0,

lambda = 10^seq(2,-2, length = 100),

standardize = TRUE,

nfolds = 5)

data.frame(lambda = model\_ridge$lambda,

cv\_mse = model\_ridge$cvm) %>%

ggplot(aes(x = lambda, y = cv\_mse)) +

geom\_point() +

geom\_line() +

geom\_vline(xintercept = model\_ridge$lambda.min, col = "deepskyblue3") +

geom\_hline(yintercept = min(model\_ridge$cvm), col = "deepskyblue3") +

scale\_x\_continuous(trans = 'log10', breaks = c(0.01, 0.1, 1, 10, 100), labels = c(0.01, 0.1, 1, 10, 100)) +

scale\_y\_continuous(labels = scales::comma\_format()) +

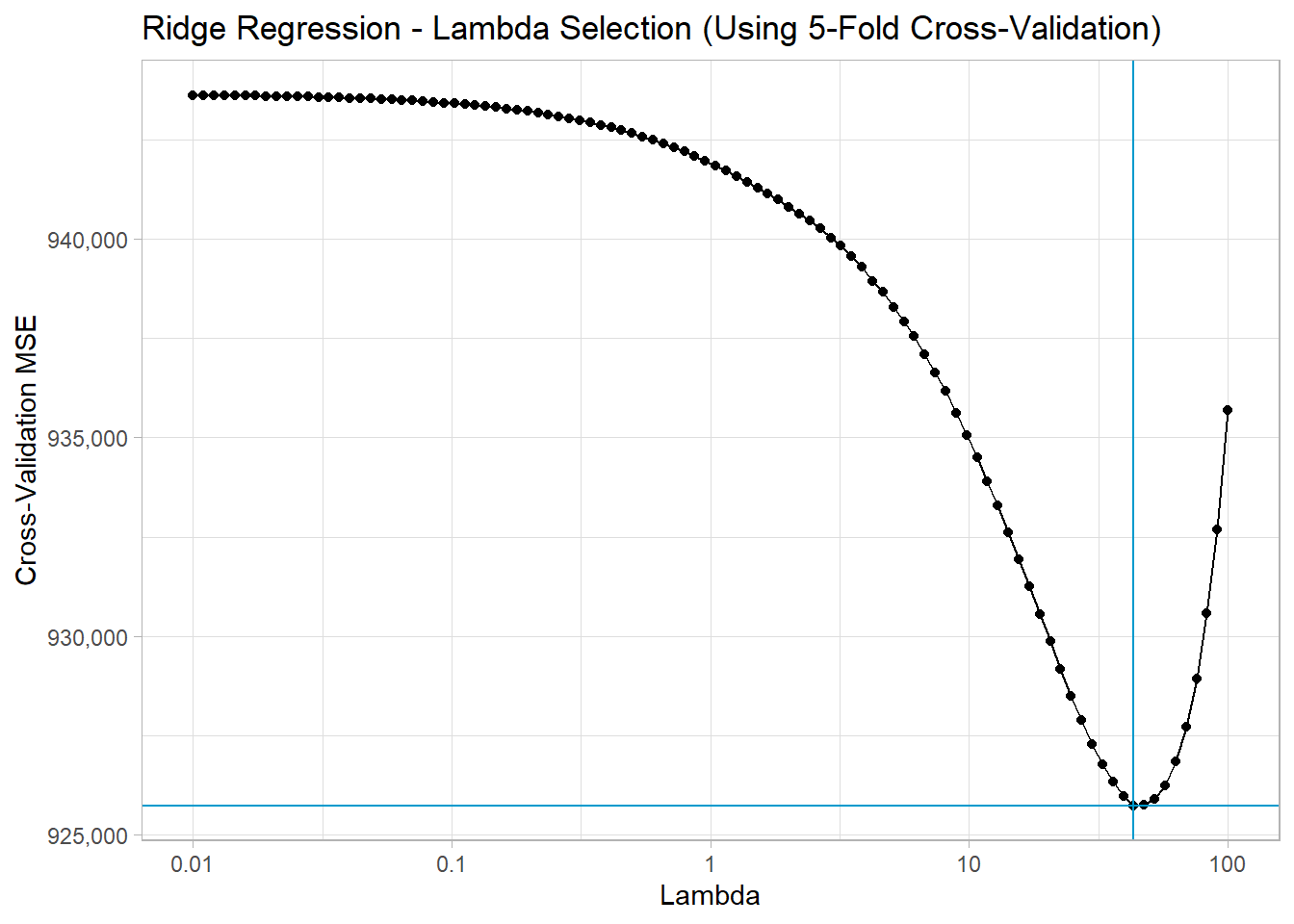
theme(legend.position = "bottom") +

labs(x = "Lambda",

y = "Cross-Validation MSE",

col = "Non-Zero Coefficients:",

title = "Ridge Regression - Lambda Selection (Using 5-Fold Cross-Validation)")



The selected value of λλ is **43.2876**. I re-fit the model and produce predictions for this value of lambda on the test data.

model\_ridge\_best <- glmnet(y = train$Apps,

x = train\_mat,

alpha = 0,

lambda = 10^seq(2,-2, length = 100))

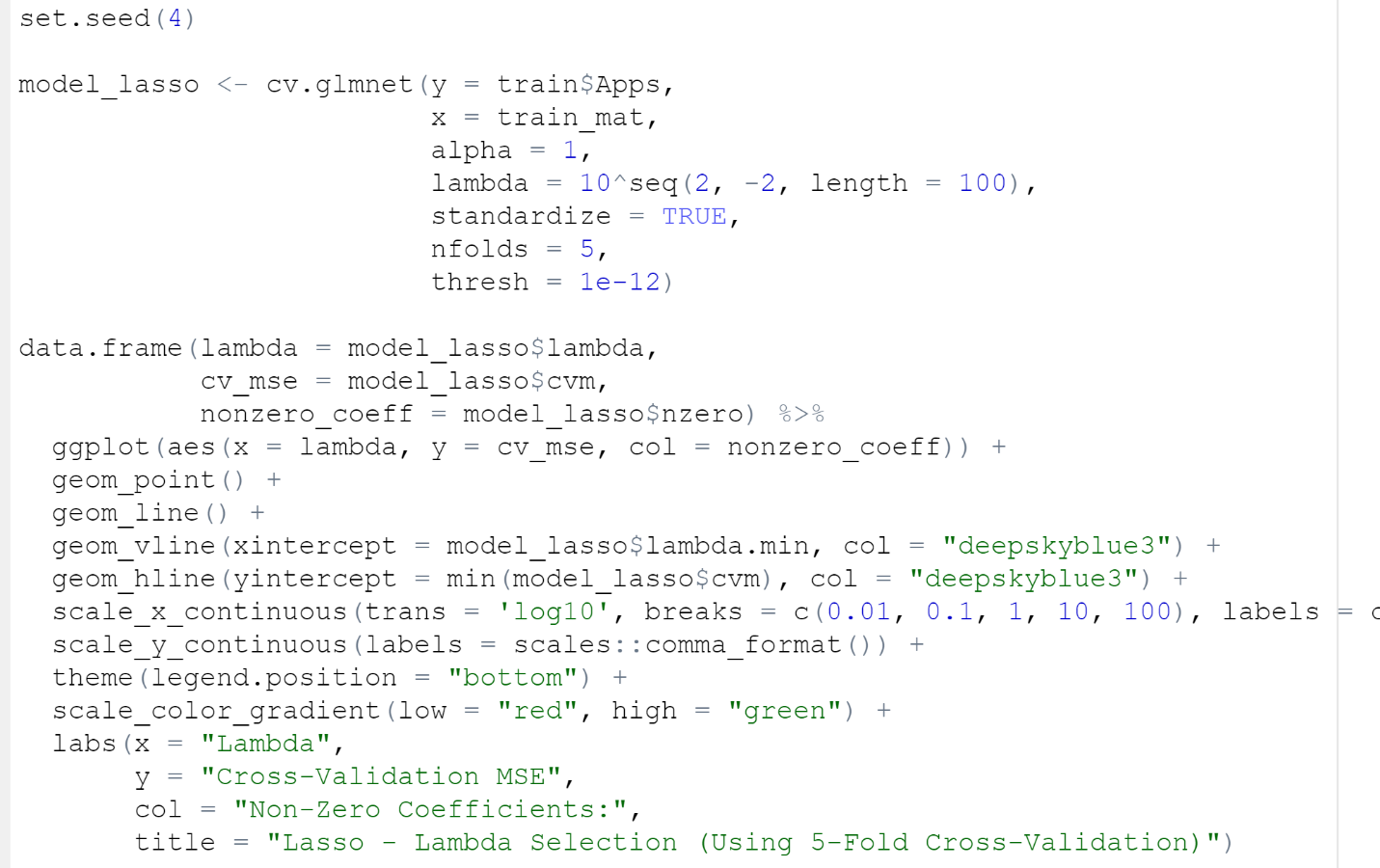
ridge\_pred <- predict(model\_ridge\_best, s = model\_ridge$lambda.min, newx = test\_mat)

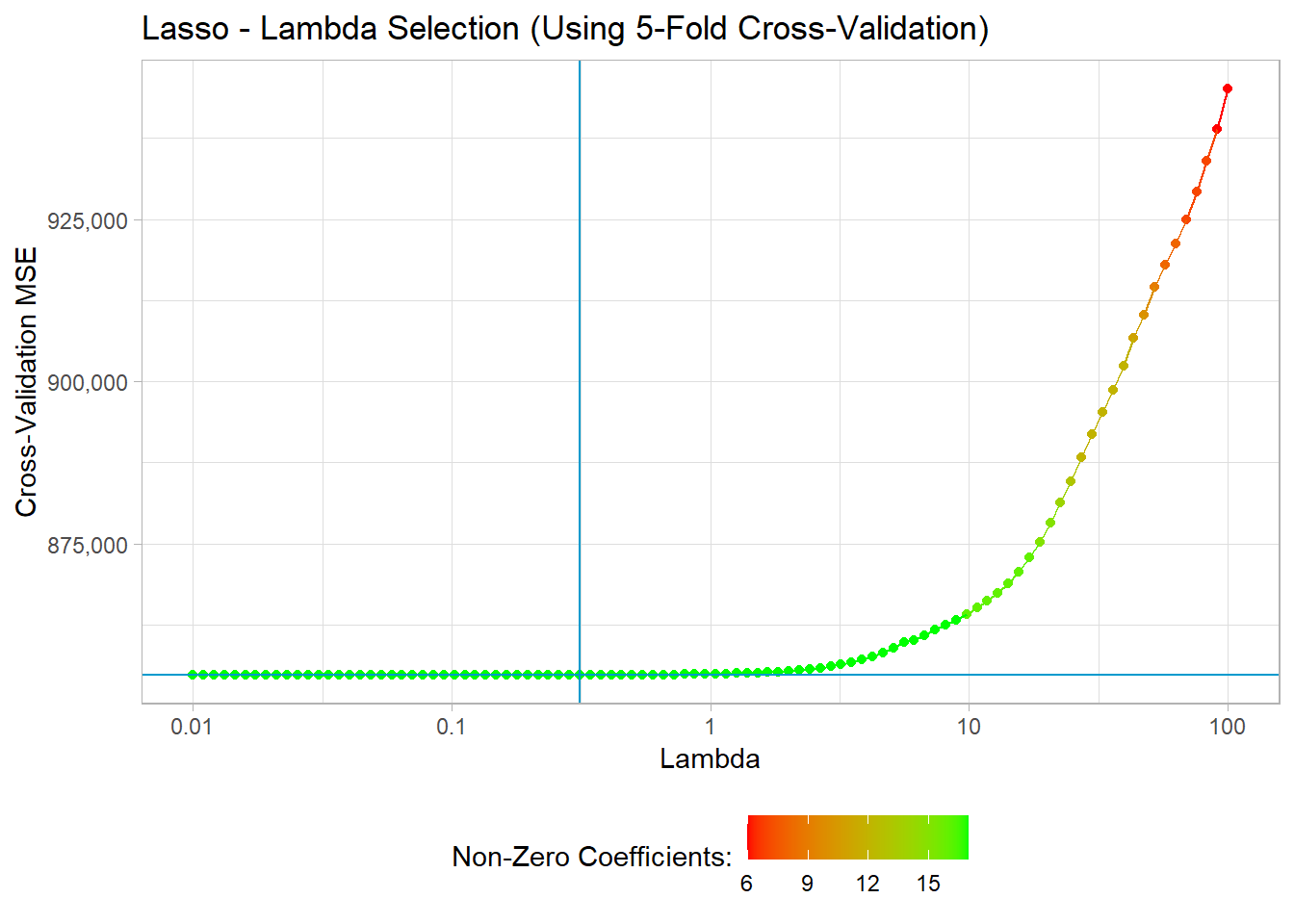
(ridge\_mse <- mean((ridge\_pred - test$Apps)^2))

2297697

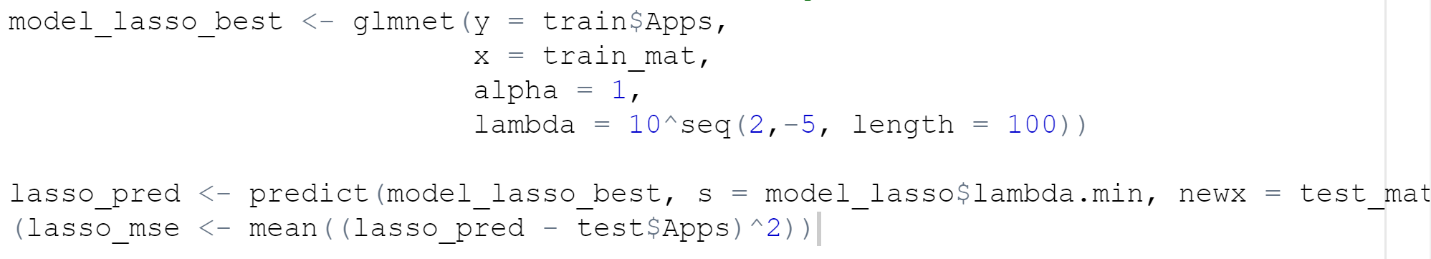
The resulting test MSE is 2297697.

d)

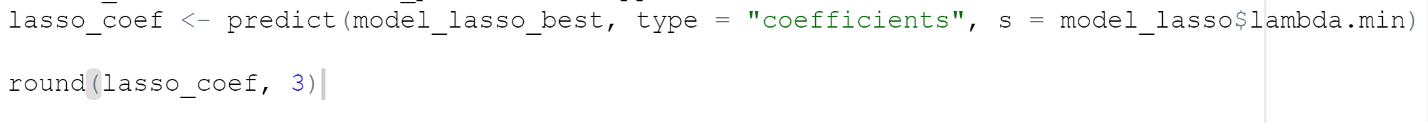




The selected value of λλ is **0.3126.**



2025346



19 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) -1162.486

Private.No 532.620

Private.Yes .

Accept 1.314

Enroll -0.402

Top10perc 50.654

Top25perc -12.190

F.Undergrad 0.073

P.Undergrad 0.039

Outstate -0.043

Room.Board 0.214

Books 0.145

Personal 0.024

PhD -8.501

Terminal -1.760

S.F.Ratio 11.042

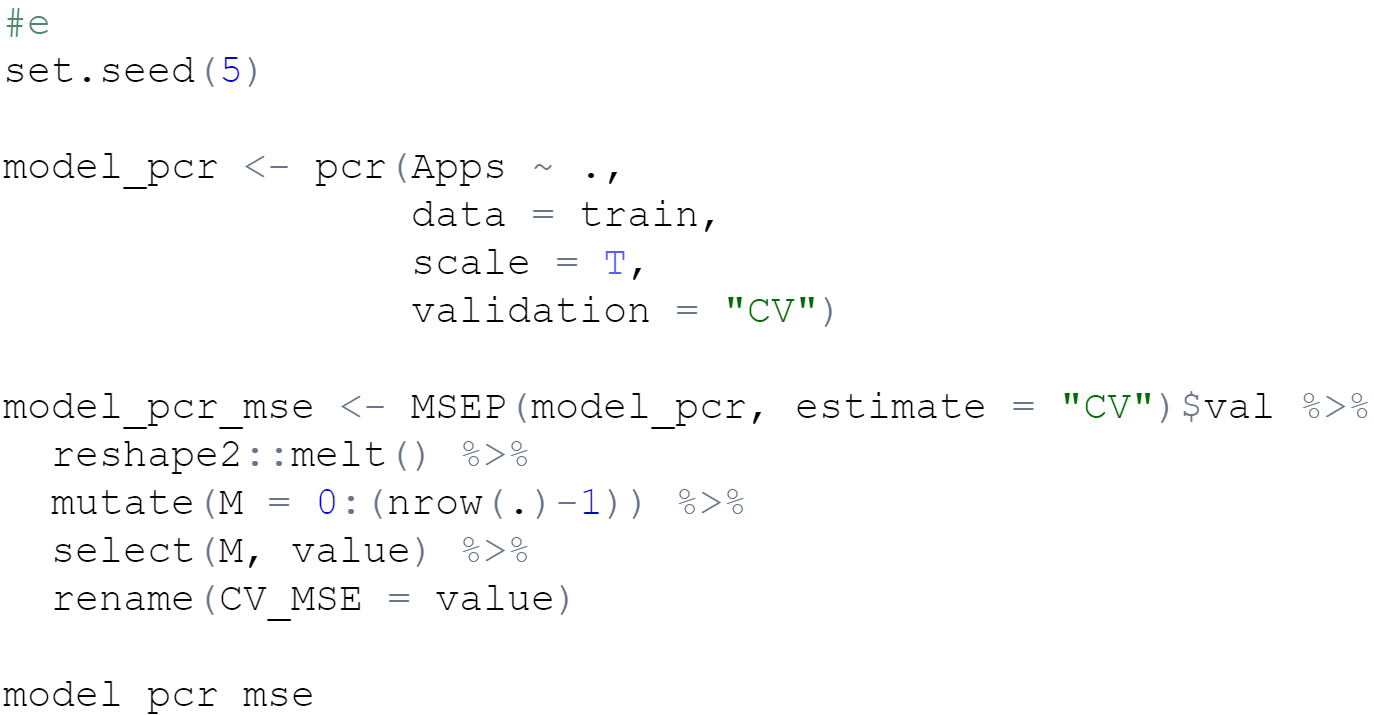
perc.alumni -7.051

Expend 0.040

Grad.Rate 5.934

We can see that 18 of the 19 coefficients are non-zero.

e)



M CV\_MSE

1 0 10884668.5

2 1 10907595.4

3 2 2569406.1

4 3 2641560.1

5 4 2567752.3

6 5 1464503.0

7 6 1400044.1

8 7 1351492.1

9 8 1303898.4

10 9 1230513.5

11 10 1220249.5

12 11 1226540.0

13 12 1238549.3

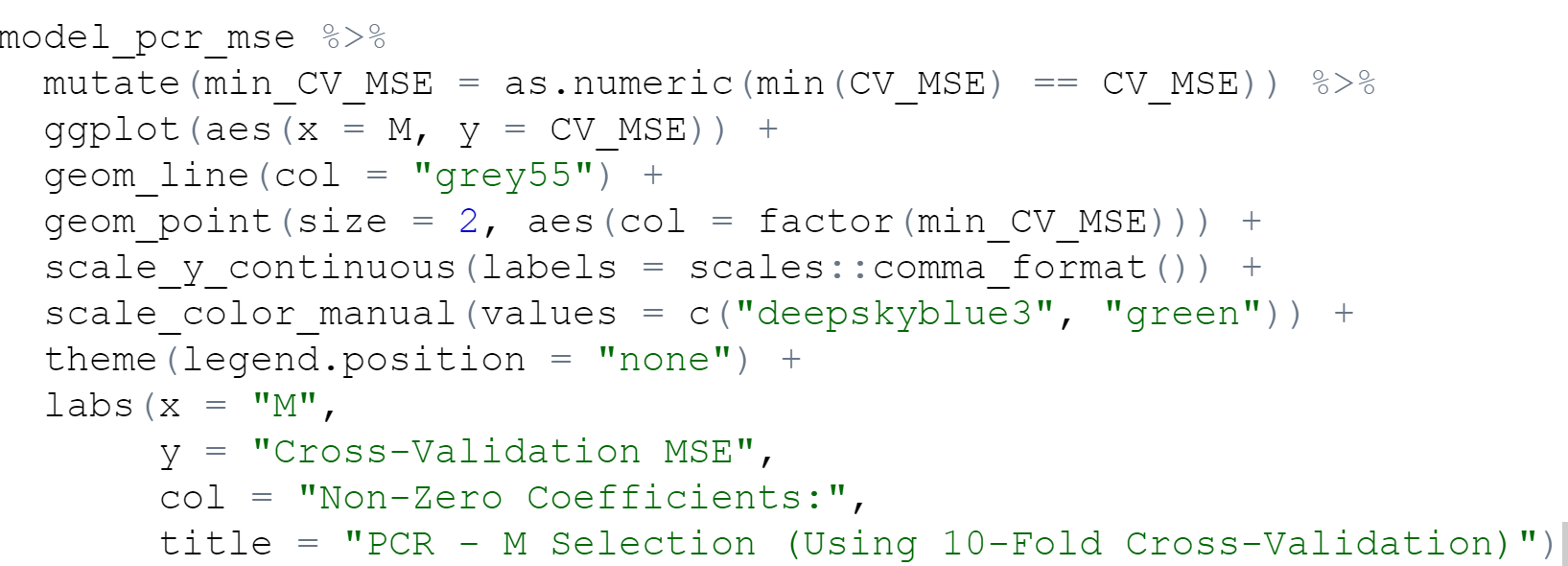
14 13 1240215.1

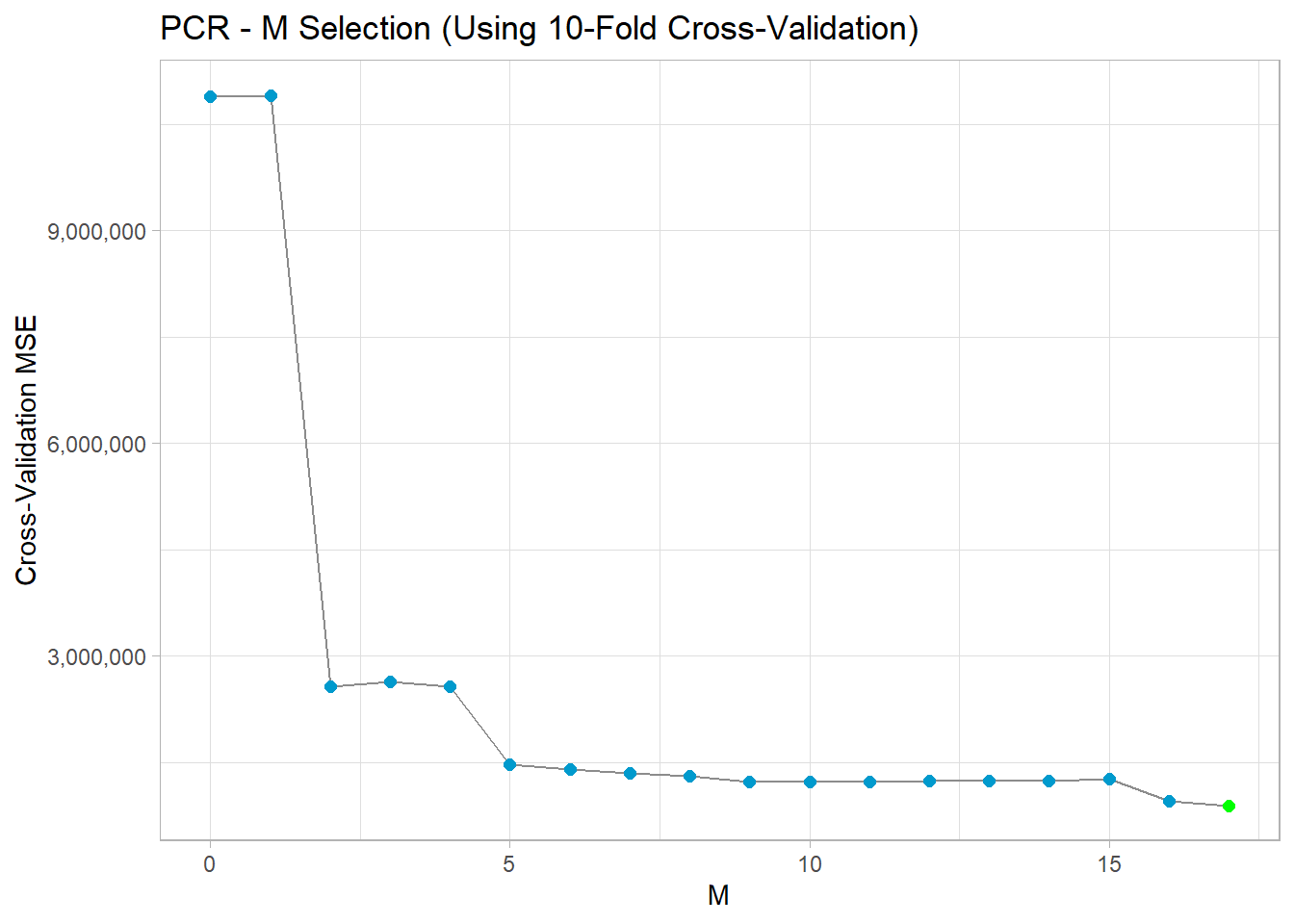
15 14 1242592.0

16 15 1263342.5

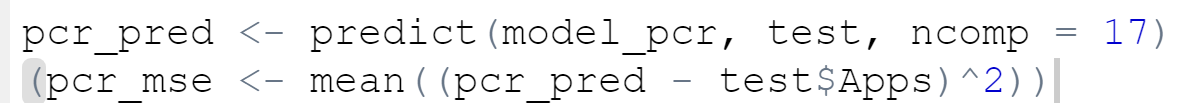
17 16 950107.6

18 17 893799.2



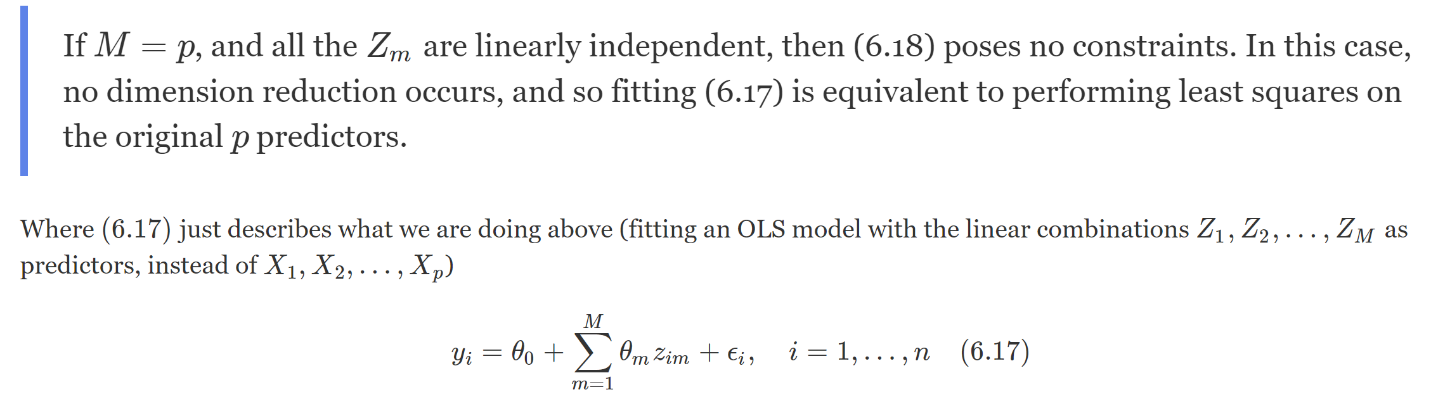


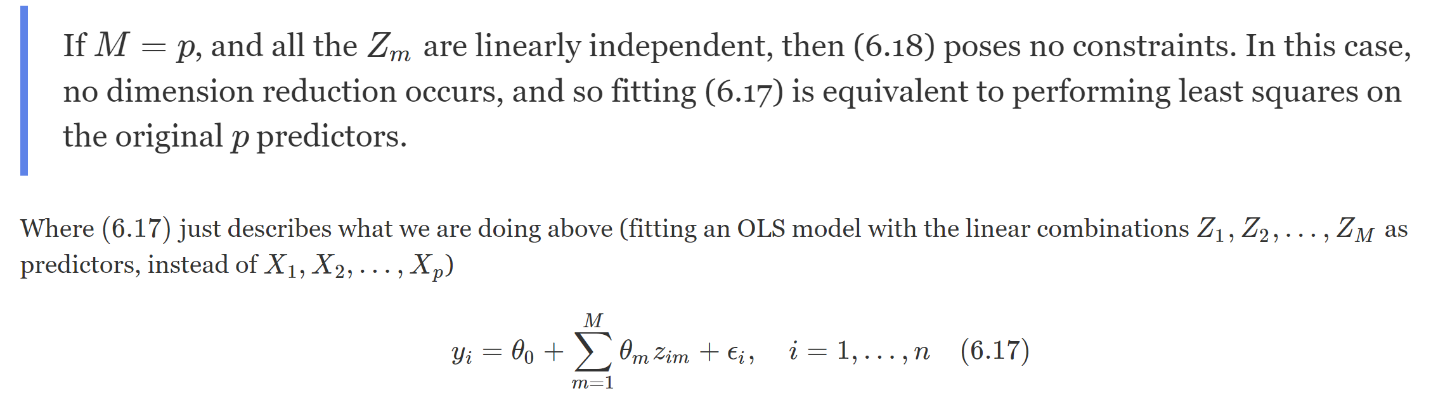
Cross-validation selected MM = 17 (so we can note that M=p=17M=p=17) - the dimensionality hasn’t been reduced at all.



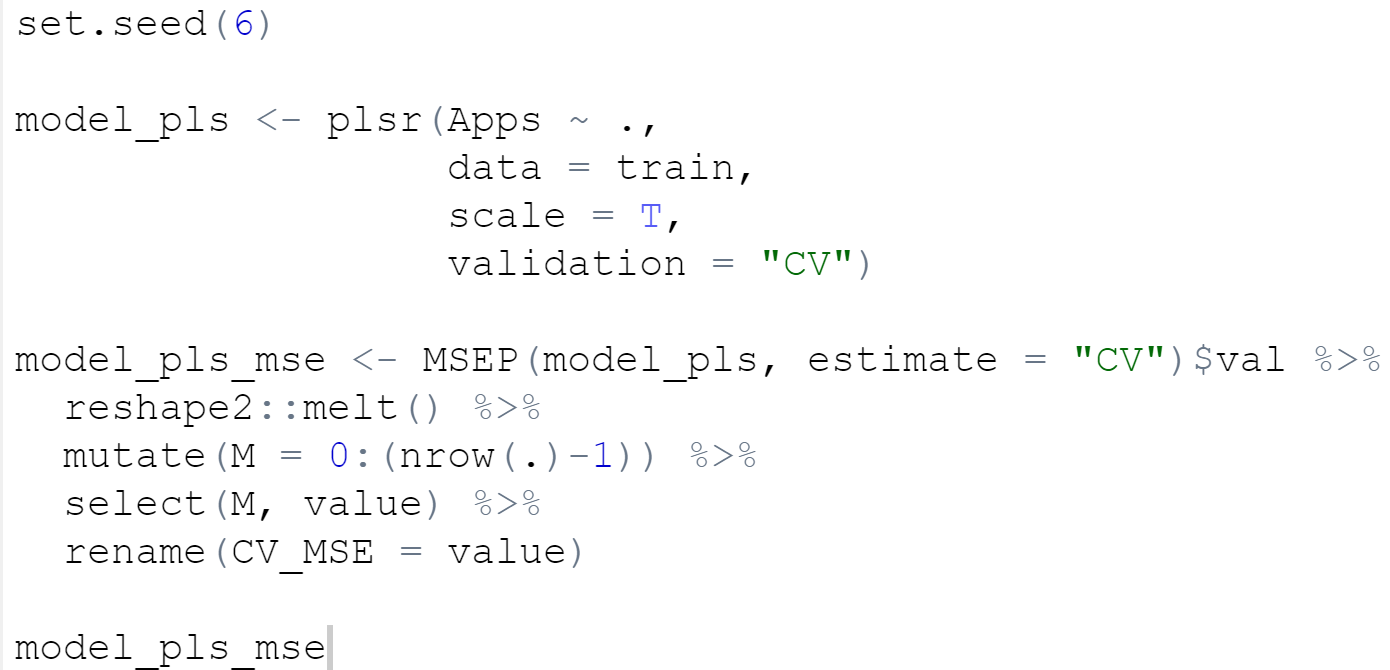
2019409

The test MSE is 2019409. I noticed here that the test MSE is identical to that obtained in part (b) from performing OLS. This is as expected from p.230 of the ISLR book, where it says:





f)



M CV\_MSE

1 0 10884668.5

2 1 2085322.2

3 2 1198711.4

4 3 1169960.5

5 4 1055530.1

6 5 951850.4

7 6 898388.2

8 7 889700.3

9 8 889619.9

10 9 884242.9

11 10 885543.9

12 11 882526.5

13 12 880744.0

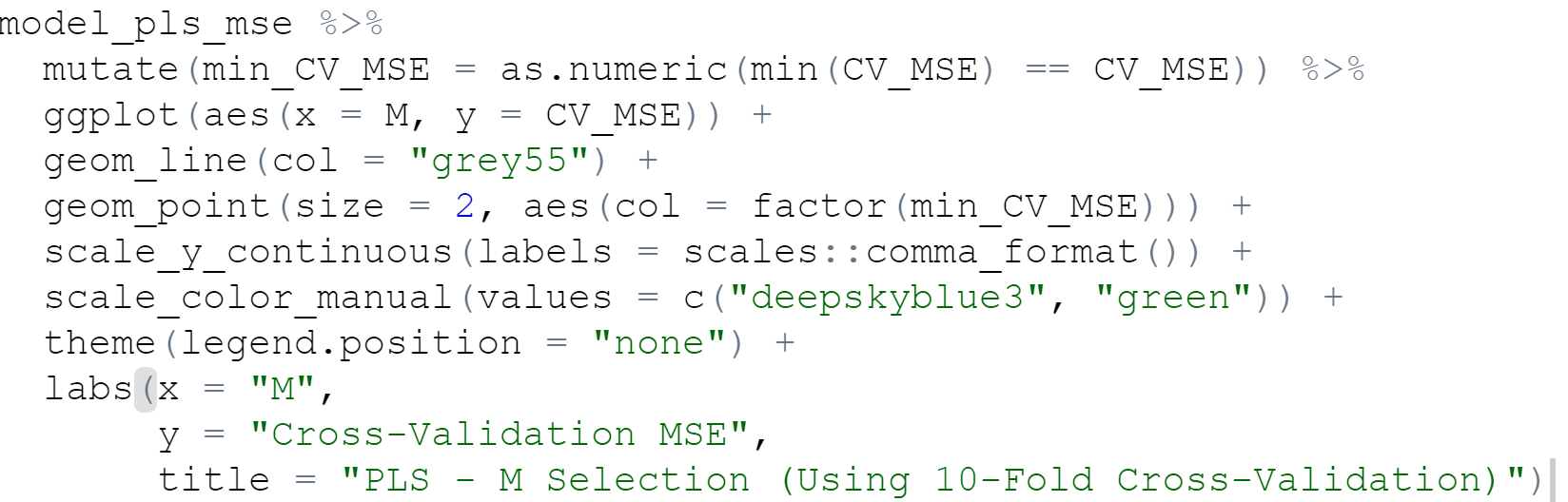
14 13 880911.3

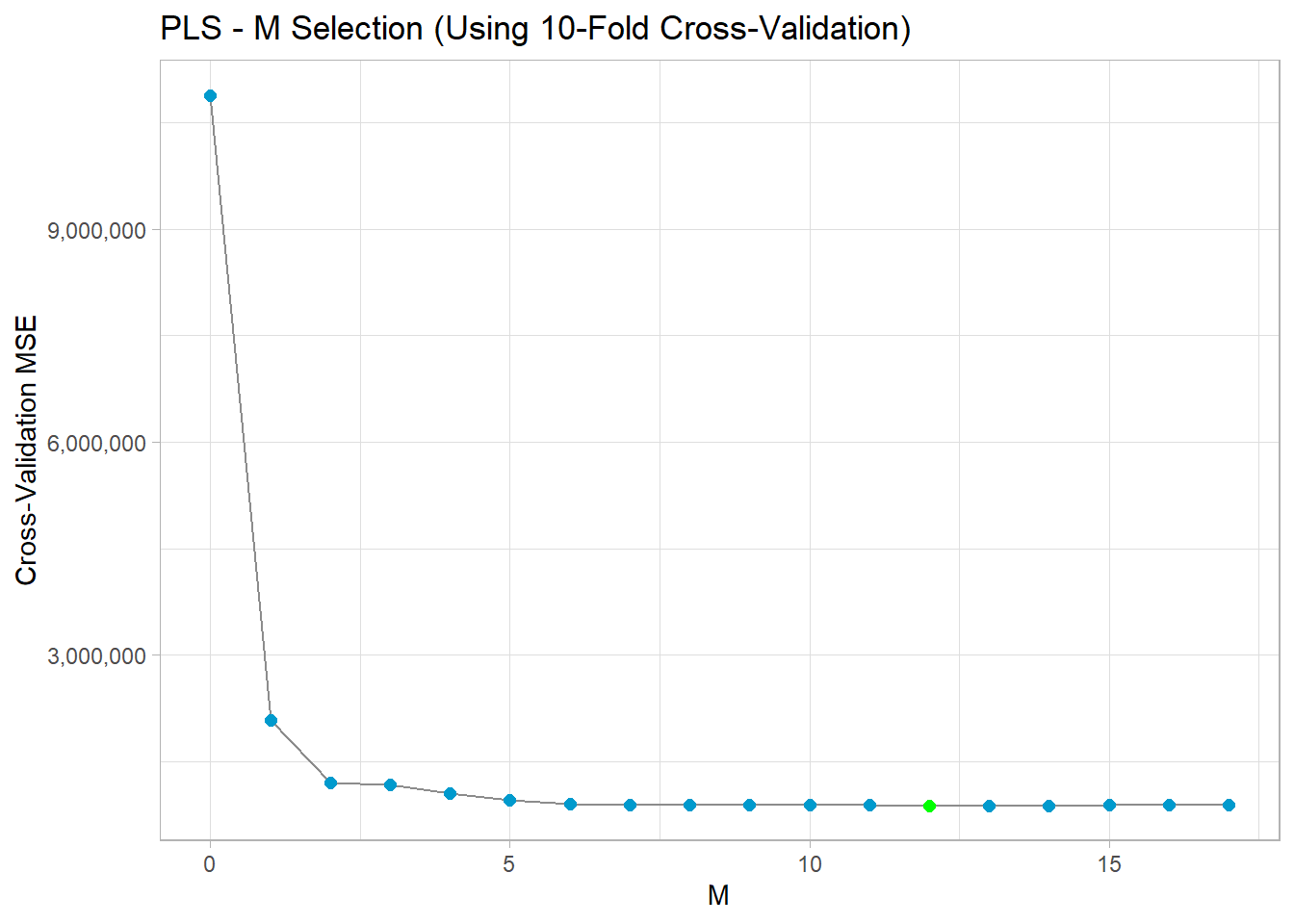
15 14 881088.5

16 15 881559.9

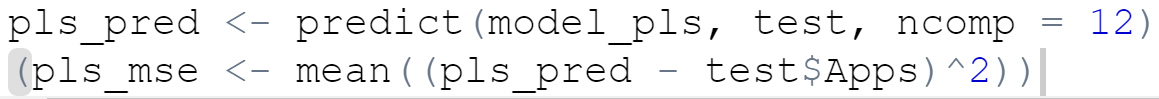
17 16 881677.0

18 17 881688.1





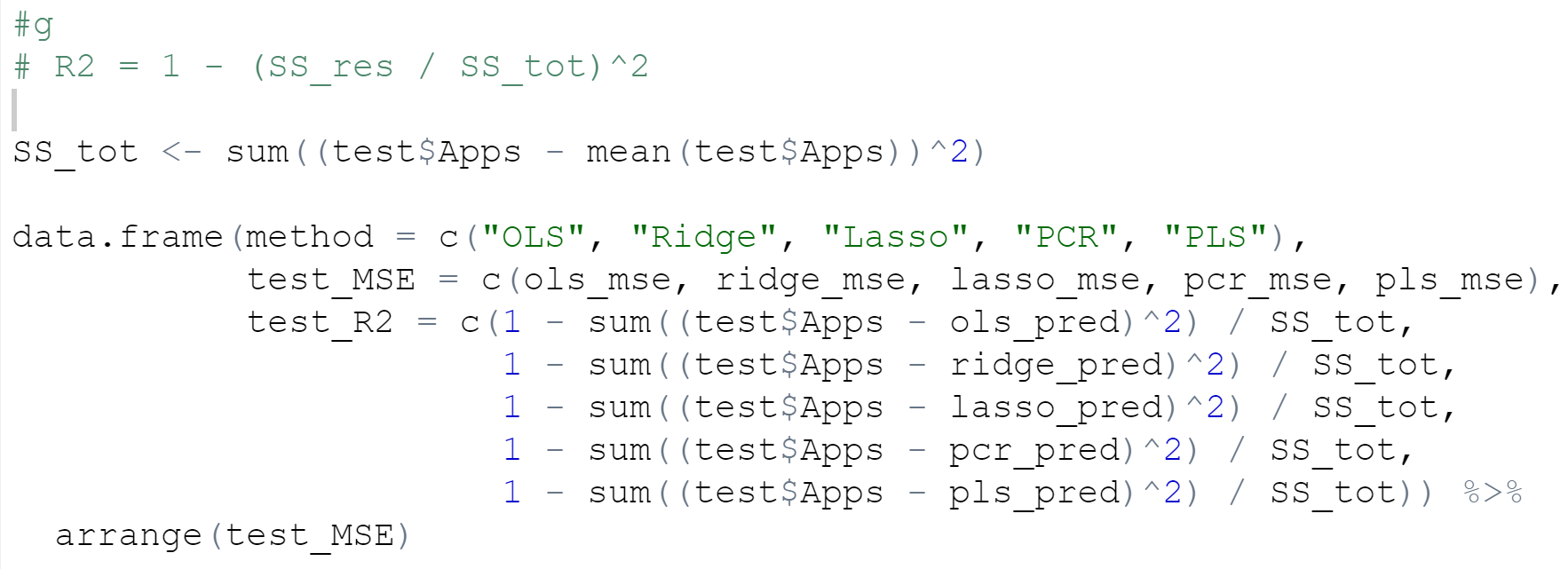
Cross-validation selected MM = 12 as the number of principal components to minimize the out-of-sample MSE.



2019361

The test MSE is 2019361.

g)



method test\_MSE test\_R2

1 PLS 2019361 0.9168871

2 PCR 2019409 0.9168851

3 OLS 2019409 0.9168851

4 Lasso 2025346 0.9166408

5 Ridge 2297697 0.9054313

There is a small difference between each of the 5 methods - for instance, the performance-boost in going from OLS to PLS is a reduction in test MSE by 0.0024%. The only model that could potentially have a meaningful performance difference was ridge regression (which performed the worst).

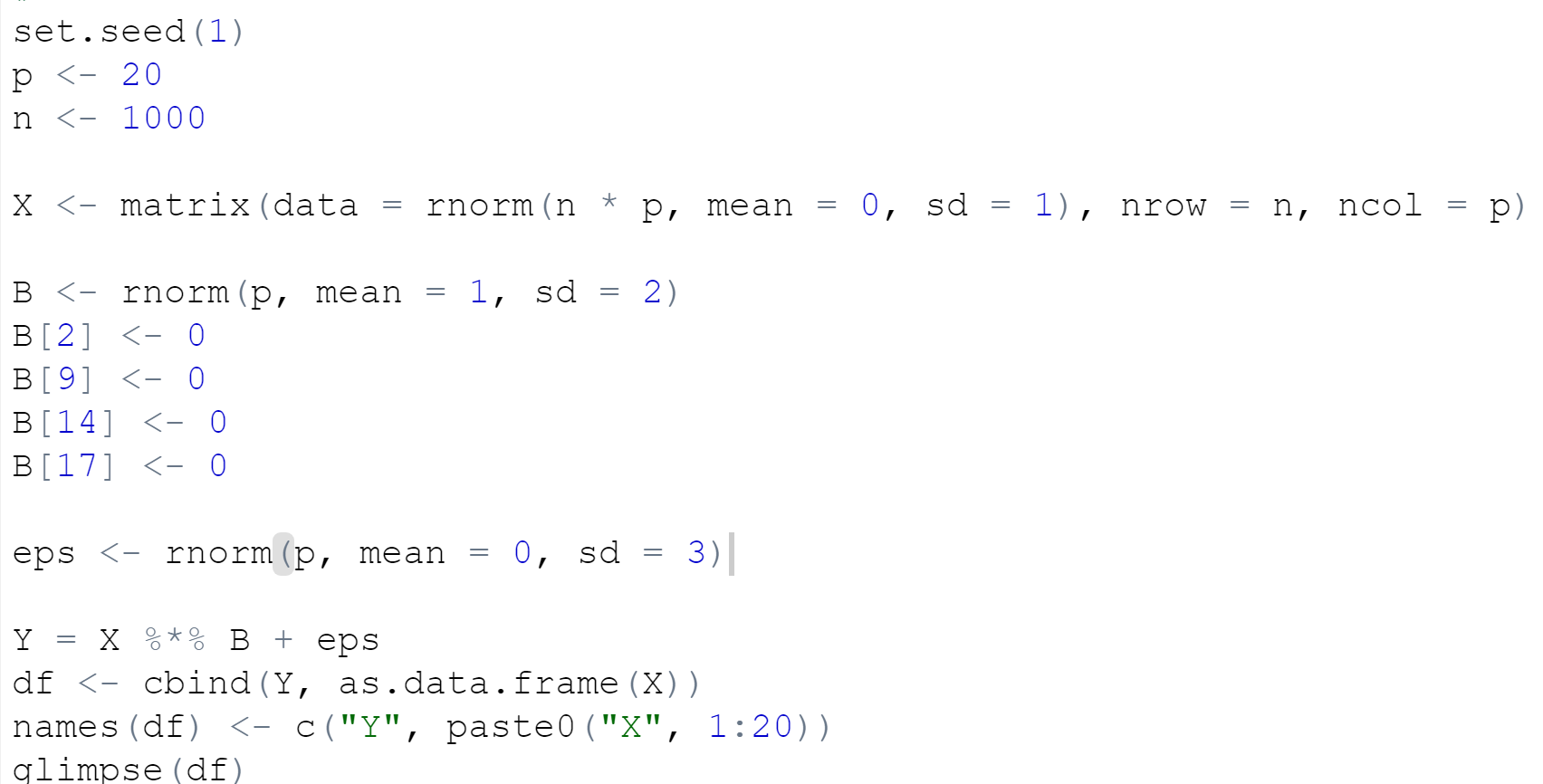
Given the size of the dataset, I have no doubt that the selected models and test performance would shift around if the random seeds were changed. Each of the models performed well, with test R2R2 values above 0.9.

10.

a) I did the design matrix X by generating a vector of length n×p=20×1,000 (sampled from a normal distribution where mean = 0 and sd = 1), and filling in X column-by-column.

Similarly, for β, I generate a two vectors of length n=1,000; β∼N(1,22) (with 4 of these subsequently overwritten to zero), and ϵ∼N(0,32).

Y is then generated by calculating Y=Xβ+ϵ.



Rows: 1,000

Columns: 21

$ Y <dbl> -7.5791363, 9.2157068, 8.2360422, 3.7628364, -6.4013460, 5.5314...

$ X1 <dbl> -0.62645381, 0.18364332, -0.83562861, 1.59528080, 0.32950777, -...

$ X2 <dbl> 1.13496509, 1.11193185, -0.87077763, 0.21073159, 0.06939565, -1...

$ X3 <dbl> -0.88614959, -1.92225490, 1.61970074, 0.51926990, -0.05584993, ...

$ X4 <dbl> 0.73911492, 0.38660873, 1.29639717, -0.80355836, -1.60262567, 0...

$ X5 <dbl> -1.13463018, 0.76455710, 0.57071014, -1.35169393, -2.02988547, ...

$ X6 <dbl> -1.5163733, 0.6291412, -1.6781940, 1.1797811, 1.1176545, -1.237...

$ X7 <dbl> -0.61882708, -1.10942196, -2.17033523, -0.03130307, -0.26039848...

$ X8 <dbl> -1.32541772, 0.95197972, 0.86000439, 1.06079031, -0.35058396, -...

$ X9 <dbl> 0.26370340, -0.82945185, -1.46163477, 1.68399018, -1.54432429, ...

$ X10 <dbl> -1.21712008, -0.94622927, 0.09140980, 0.70135127, 0.67342236, 1...

$ X11 <dbl> -0.80433160, -1.05652565, -1.03539578, -1.18556035, -0.50043951...

$ X12 <dbl> -1.411521883, 1.083869657, 1.170222351, 0.294754540, -0.5544276...

$ X13 <dbl> -0.93910663, 1.39366493, 1.62581486, 0.40900106, -0.09255856, 0...

$ X14 <dbl> 0.2264537, -0.8185942, -0.8471526, -1.9843326, -0.8127788, 1.46...

$ X15 <dbl> 0.5232667, 0.9935537, 0.2737370, -0.6949193, -0.7180502, -0.101...

$ X16 <dbl> -0.21390898, -0.10672328, -0.46458931, -0.68427247, -0.79080075...

$ X17 <dbl> 0.85763410, -1.62539515, -0.23427831, -1.03265445, -1.14114122,...

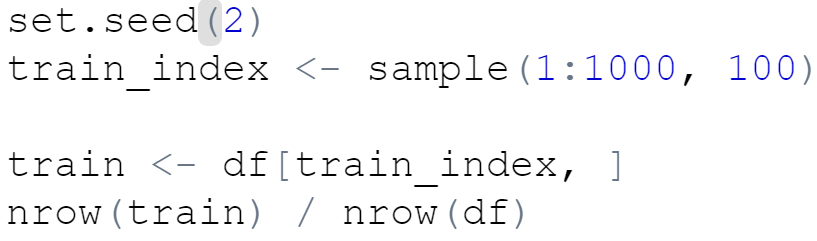
$ X18 <dbl> 1.0496171412, 0.2903237344, 1.2421262227, -0.6850857039, -0.667...

$ X19 <dbl> 0.95140989, 0.45709866, -0.35869346, -1.04586136, 0.30753453, 1...

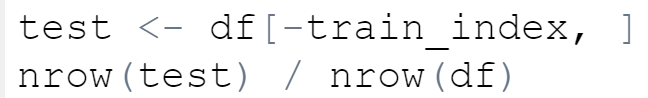
$ X20 <dbl> -2.07771241, -0.45446091, -0.16555991, 0.89765209, -0.02948916,...

b)

The “train” contains 10% of the total observations:

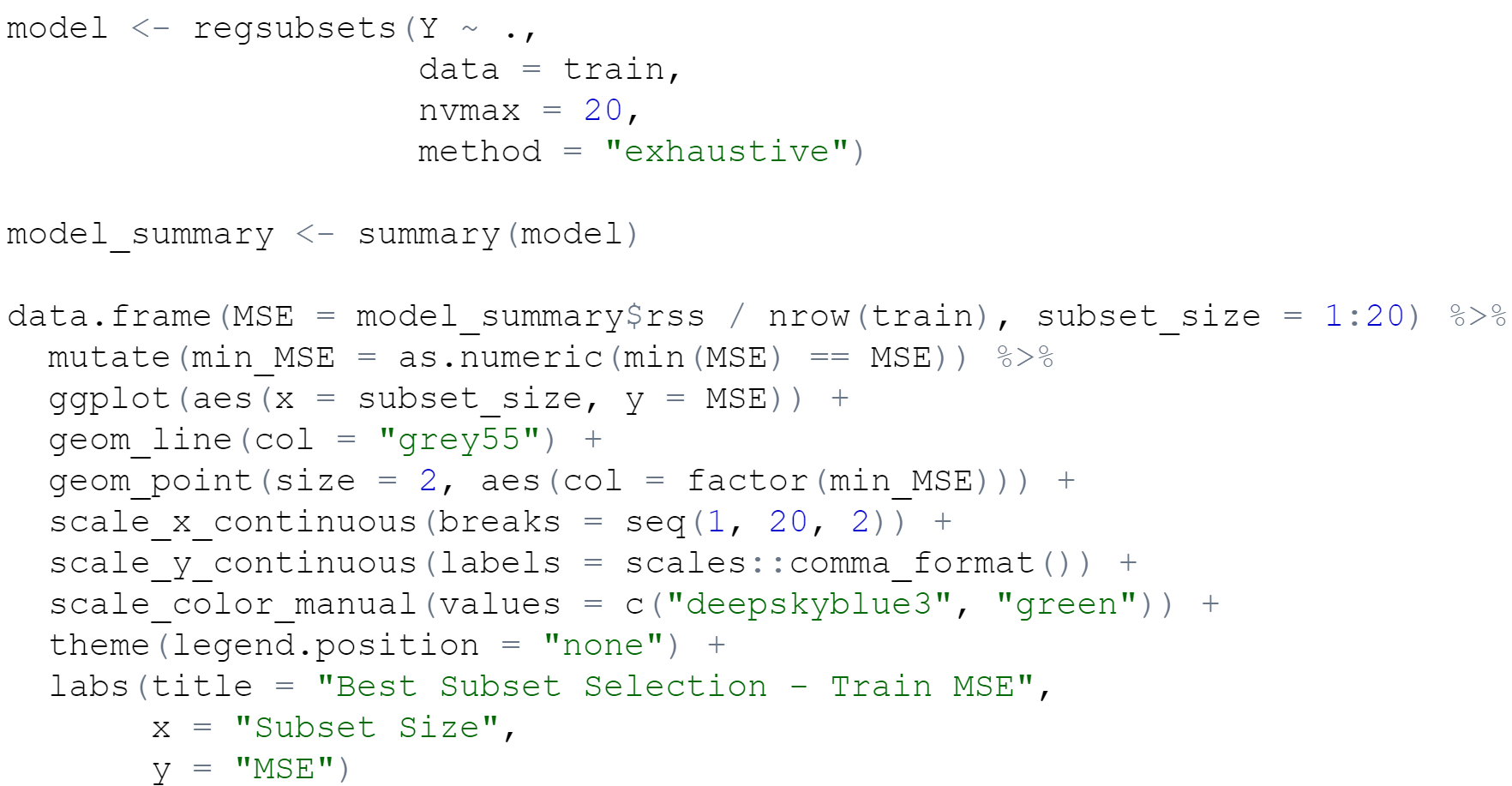


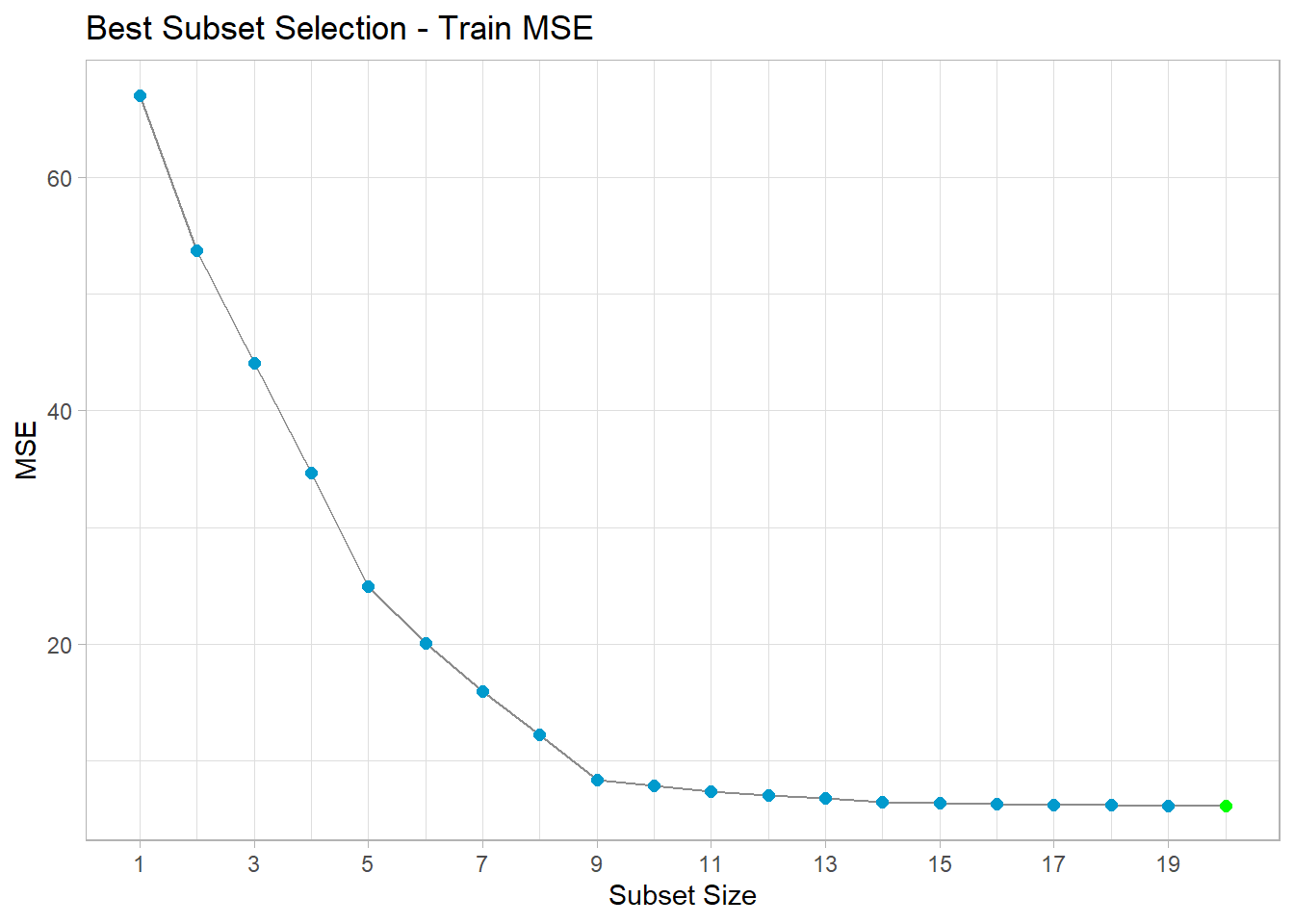
[1] 0.1



[1] 0.9

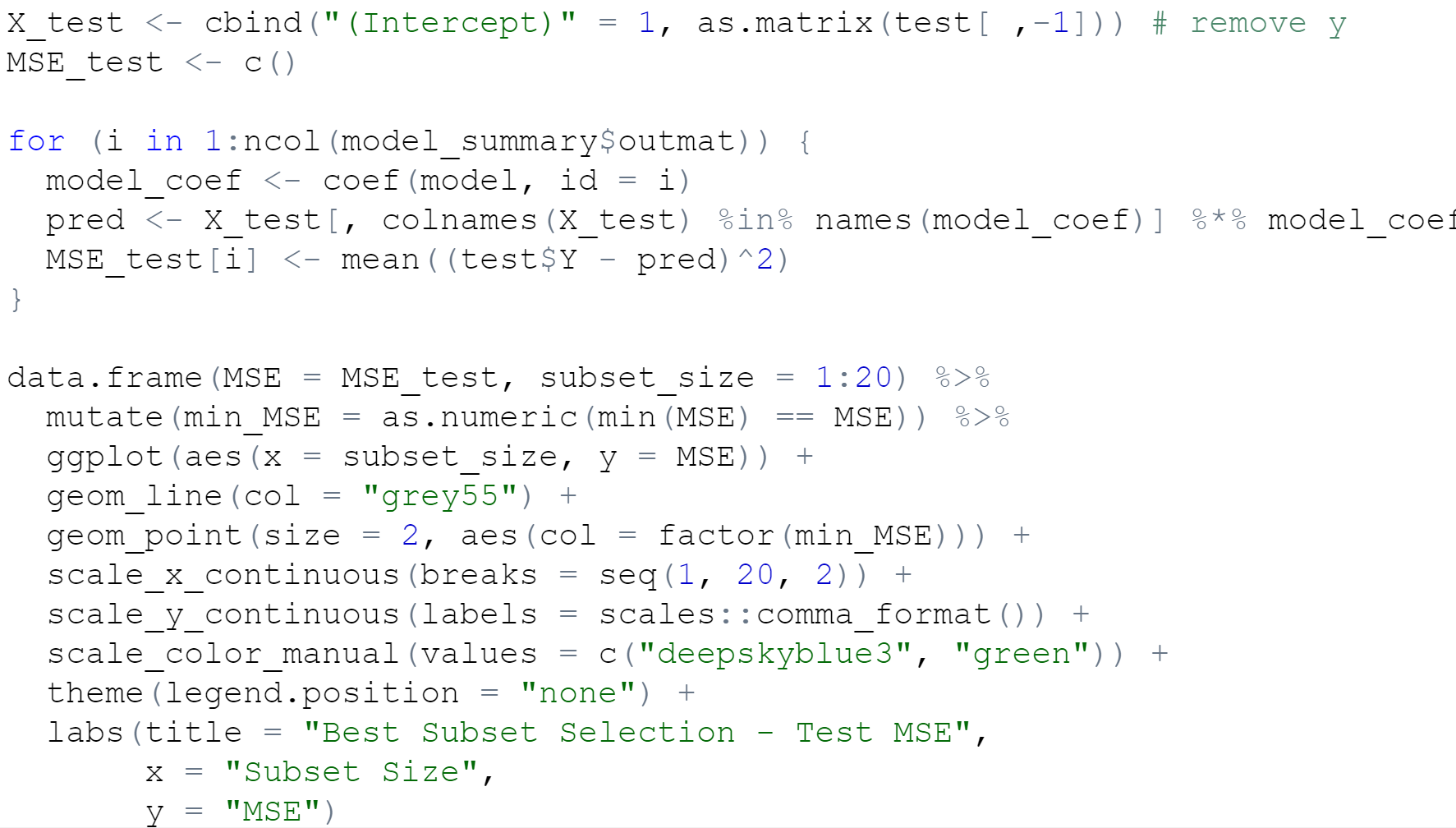
c)

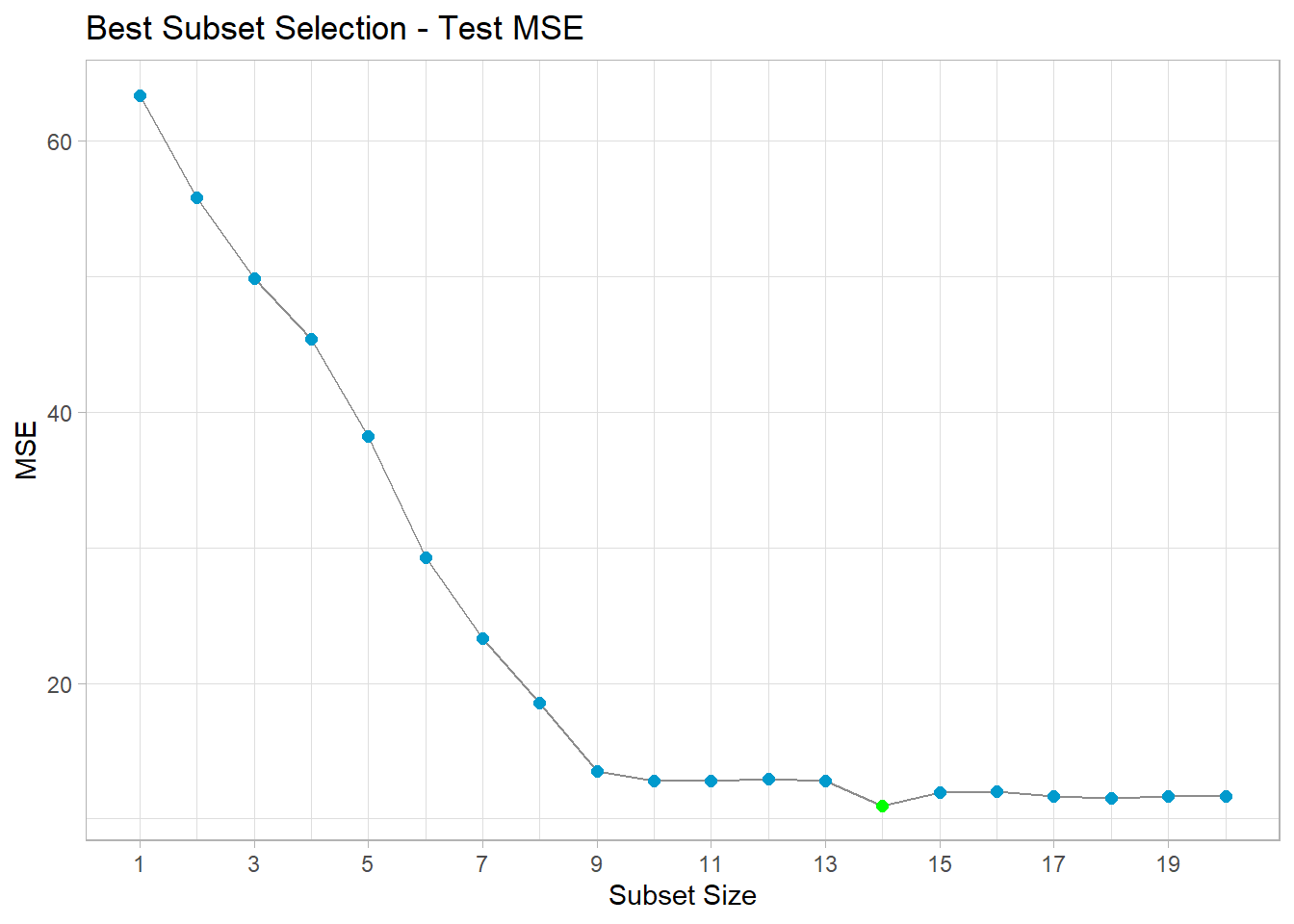




This graph shows as subset size increases, the MSE (of train) decreases.

d)





This graph shows as the subset size increases, the MSE (of test) decreases.

e)

The test MSE is minimized for the 14-variable model. The test MSE graph shows as the number of variables in the subset initially increase, then after a certain point, larger subsets make very little difference. In practice, for very large datasets I would probably choose the 14-variable model, and for very small datasets such as this, I would choose the 9-variable model.