Homework 5 - Coding

John L Kaspers

June 10, 2020

Warning

Warning: If you get an error when running the code, make sure that you have the necessary packages installed. You can install these packages by running the following code in base R (not RStudio or RMarkdown).

> install.packages("alr4")
> install.packages("cvTools")
> install.packages("glmnet")
> install.packages("sandwich")

Model Comparison (5 points)

Use the `Puromycin` data set built into R. Further information on the data set can be found here.

For this problem, you will run through various methods of model comparison - Adjusted R-squared, AIC, LRT, and LOOCV.

1. Create two models that predict `rate`. The first model will use `conc` as the predictor. The second model will use `conc` and `state` as predictors. Do not include any interactions of other transformations. Run each of your models through the `summary()` function.

```
data(Puromycin)

lm.predict.rate1 <- lm(rate ~ conc, data = Puromycin)

lm.predict.rate2 <- lm(rate ~ conc + state, data = Puromycin)

summary(lm.predict.rate1)
```

```
##
## Call:
## lm(formula = rate ~ conc, data = Puromycin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.861 -15.247  -2.861  15.686  48.054
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    93.92       8.00   11.74 1.09e-10 ***
## conc          105.40      16.92    6.23 3.53e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.82 on 21 degrees of freedom
## Multiple R-squared:  0.6489, Adjusted R-squared:  0.6322
## F-statistic: 38.81 on 1 and 21 DF,  p-value: 3.526e-06
```

```
summary(lm.predict.rate2)
```

```
##
## Call:
## lm(formula = rate ~ conc + state, data = Puromycin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.381 -16.502   4.268  21.346  37.452
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     106.338      9.413  11.296 3.92e-10 ***
## conc            102.160     15.721   6.498 2.46e-06 ***
## stateuntreated  -23.844     11.177  -2.133   0.0455 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.65 on 20 degrees of freedom
## Multiple R-squared:  0.714,  Adjusted R-squared:  0.6854
## F-statistic: 24.96 on 2 and 20 DF,  p-value: 3.667e-06
```

Which model is preferred according to adjusted R-squared?

According to adjusted R-squared, the model with conc and state as predictors is preferred because it has a slightly larger adjusted R-squared (0.6854 compared to 0.6322).

2. Use the `AIC()` function to calculate the AIC for each model.

```
AIC(lm.predict.rate1)
```

```
## [1] 223.7834
```

```
AIC(lm.predict.rate2)
```

```
## [1] 221.0681
```

Which model is preferred according to AIC?

Because a lower AIC represents a better fitting model, model 2 (the model with conc and state as predictors) is preferred according to AIC because model 2 has a lower AIC (221.0681 compared to model 1's 223.7834).

3. Run a likelihood ratio test between the two models.

```
anova(lm.predict.rate1, lm.predict.rate2, test = "LRT")
```

```
## Analysis of Variance Table
##
## Model 1: rate ~ conc
## Model 2: rate ~ conc + state
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1     21 17439
## 2     20 14206  1    3232.5   0.0329 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Which model is preferred according to the LRT?

According to LRT, the results are statistically significant as shown through the p-value of 0.0329, which is statistically significant at the 5% level so we should improve model 1 by adding 'state', thus model 2 is preferred according to the LRT.

4. Use the `cvFit` function within the `cvTools` package to run LOOCV for both models.

```
cvFit(lm.predict.rate1, data = Puromycin, y = Puromycin$rate, cost = rmspe, K = nrow(Puromycin))
```

```
## Leave-one-out CV results:
##       CV
## 30.66056
```

```
cvFit(lm.predict.rate2, data = Puromycin, y = Puromycin$rate, cost = rmspe, K = nrow(Puromycin))
```

```
## Leave-one-out CV results:
##       CV
## 29.01072
```

Which model is preferred according to LOOCV?

According to LOOCV, in which we want a lower RMSPE for a better fitting model, lm.predict.rate2 is marginally preferred because it has a slightly lower RMSPE (29.01072 compared to model1's 30.66056).

Lasso (5 points)

Use the `swiss` data set built into R. Further information on the data set can be found here.

For this problem, you will use Lasso regression to predict `Fertility` based on all other predictors. You will need to utilize the `glmnet()` and `cv.glmnet()` functions within the `glmnet` package. See the Lecture Notes for details.

1. Create the design matrix using the `model.matrix()` function. This is used as one of the arguments for `glmnet()` when running Lasso.

```
data(swiss)
model.matrix(Fertility ~ ., data = swiss)
```

```
##              (Intercept) Agriculture Examination Education Catholic
## Courtelary             1        17.0          15        12     9.96
## Delemont               1        45.1           6         9    84.84
## Franches-Mnt           1        39.7           5         5    93.40
## Moutier                1        36.5          12         7    33.77
## Neuveville             1        43.5          17        15     5.16
## Porrentruy             1        35.3           9         7    90.57
## Broye                  1        70.2          16         7    92.85
## Glane                  1        67.8          14         8    97.16
## Gruyere                1        53.3          12         7    97.67
## Sarine                 1        45.2          16        13    91.38
## Veveyse                1        64.5          14         6    98.61
## Aigle                  1        62.0          21        12     8.52
## Aubonne                1        67.5          14         7     2.27
## Avenches               1        60.7          19        12     4.43
## Cossonay               1        69.3          22         5     2.82
## Echallens              1        72.6          18         2    24.20
## Grandson               1        34.0          17         8     3.30
## Lausanne               1        19.4          26        28    12.11
## La Vallee              1        15.2          31        20     2.15
## Lavaux                 1        73.0          19         9     2.84
## Morges                 1        59.8          22        10     5.23
## Moudon                 1        55.1          14         3     4.52
## Nyone                  1        50.9          22        12    15.14
## Orbe                   1        54.1          20         6     4.20
## Oron                   1        71.2          12         1     2.40
## Payerne                1        58.1          14         8     5.23
## Paysd'enhaut           1        63.5           6         3     2.56
## Rolle                  1        60.8          16        10     7.72
## Vevey                  1        26.8          25        19    18.46
## Yverdon                1        49.5          15         8     6.10
## Conthey                1        85.9           3         2    99.71
## Entremont              1        84.9           7         6    99.68
## Herens                 1        89.7           5         2   100.00
## Martigwy               1        78.2          12         6    98.96
## Monthey                1        64.9           7         3    98.22
## St Maurice             1        75.9           9         9    99.06
## Sierre                 1        84.6           3         3    99.46
## Sion                   1        63.1          13        13    96.83
## Boudry                 1        38.4          26        12     5.62
## La Chauxdfnd           1         7.7          29        11    13.79
## Le Locle               1        16.7          22        13    11.22
## Neuchatel              1        17.6          35        32    16.92
## Val de Ruz             1        37.6          15         7     4.97
## ValdeTravers           1        18.7          25         7     8.65
## V. De Geneve           1         1.2          37        53    42.34
## Rive Droite            1        46.6          16        29    50.43
## Rive Gauche            1        27.7          22        29    58.33
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
## Moutier                  20.3
## Neuveville               20.6
## Porrentruy               26.6
## Broye                    23.6
## Glane                    24.9
## Gruyere                  21.0
## Sarine                   24.4
## Veveyse                  24.5
## Aigle                    16.5
## Aubonne                  19.1
## Avenches                 22.7
## Cossonay                 18.7
## Echallens                21.2
## Grandson                 20.0
## Lausanne                 20.2
## La Vallee                10.8
## Lavaux                   20.0
## Morges                   18.0
## Moudon                   22.4
## Nyone                    16.7
## Orbe                     15.3
## Oron                     21.0
## Payerne                  23.8
## Paysd'enhaut             18.0
## Rolle                    16.3
## Vevey                    20.9
## Yverdon                  22.5
## Conthey                  15.1
## Entremont                19.8
## Herens                   18.3
## Martigwy                 19.4
## Monthey                  20.2
## St Maurice               17.8
## Sierre                   16.3
## Sion                     18.1
## Boudry                   20.3
## La Chauxdfnd             20.5
## Le Locle                 18.9
## Neuchatel                23.0
## Val de Ruz               20.0
## ValdeTravers             19.5
## V. De Geneve             18.0
## Rive Droite              18.2
## Rive Gauche              19.3
## attr(,"assign")
## [1] 0 1 2 3 4 5
```

2. Run Lasso using different values of λ to determine which variable is the first to have its estimated slope coefficient go to 0. This may take multiple iterations. Remember to set the value of α to 1, as we did in the Lecture Notes.

```
library(glmnet)

lasso.model <- glmnet(model.matrix(Fertility ~ ., data = swiss), swiss$Fertility, alpha = 1)
coef(lasso.model, c(0, 0.1, 1, 1.05, 1.06, 10, 60, 100))
```

```
## 7 x 8 sparse Matrix of class "dgCMatrix"
##                           1          2             3             4
## (Intercept)      66.6423597 65.7665041 55.8153858040 56.0341537090
## (Intercept)       .          .          .             .
## Agriculture      -0.1678682 -0.1544567 -0.0007866263 -0.0001253665
## Examination      -0.2558537 -0.2467932 -0.1407473777 -0.1417288911
## Education        -0.8640181 -0.8430649 -0.6021498969 -0.5964469614
## Catholic          0.1031075  0.1000933  0.0653707291  0.0642655815
## Infant.Mortality  1.0764301  1.0734722  1.0334226904  1.0207250455
##                            5        6        7        8
## (Intercept)      56.07860107 70.14255 70.14255 70.14255
## (Intercept)       .          .        .        .
## Agriculture       .          .        .        .
## Examination      -0.14193272 .        .        .
## Education        -0.59531230 .        .        .
## Catholic          0.06404512 .        .        .
## Infant.Mortality  1.01817652 .        .        .
```

Which variable was the first to have its estimated slope coefficient go to 0?

Agriculture was the first variable to have its estimated slope coefficient go to 0.

3. At what (approximate) value of λ do **all** of the estimated slope coefficients go to 0? An integer value is fine here!

```
cv.glmnet(model.matrix(Fertility ~ ., data = swiss), swiss$Fertility, alpha = 1)
```

```
##
## Call:  cv.glmnet(x = model.matrix(Fertility ~ ., data = swiss), y = swiss$Fertility,      alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Measure    SE Nonzero
## min 0.0256   59.66  9.30       5
## 1se 1.4006   68.58 11.16       4
```

60

4. Use the `cv.glmnet()` function to determine the optimal value for λ.

```
cv.glmnet(model.matrix(Fertility ~ ., data = swiss), swiss$Fertility, alpha = 1)$lambda.min
```

```
## [1] 0.04917668
```

What value of λ is returned? Note: This will vary from trial-to-trial (that's okay!).

0.04917668

5. Run lasso regression again using your optimal λ from Part 4.

```
coef(lasso.model, 0.04917668)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                           1
## (Intercept)      66.3499116
## (Intercept)       .
## Agriculture      -0.1633465
## Examination      -0.2530268
## Education        -0.8568413
## Catholic          0.1020699
## Infant.Mortality  1.0754606
```

What variables are in the model?

Agriculture, Examination, Education, Catholic and Infant.Mortality.

Variances (12 points)

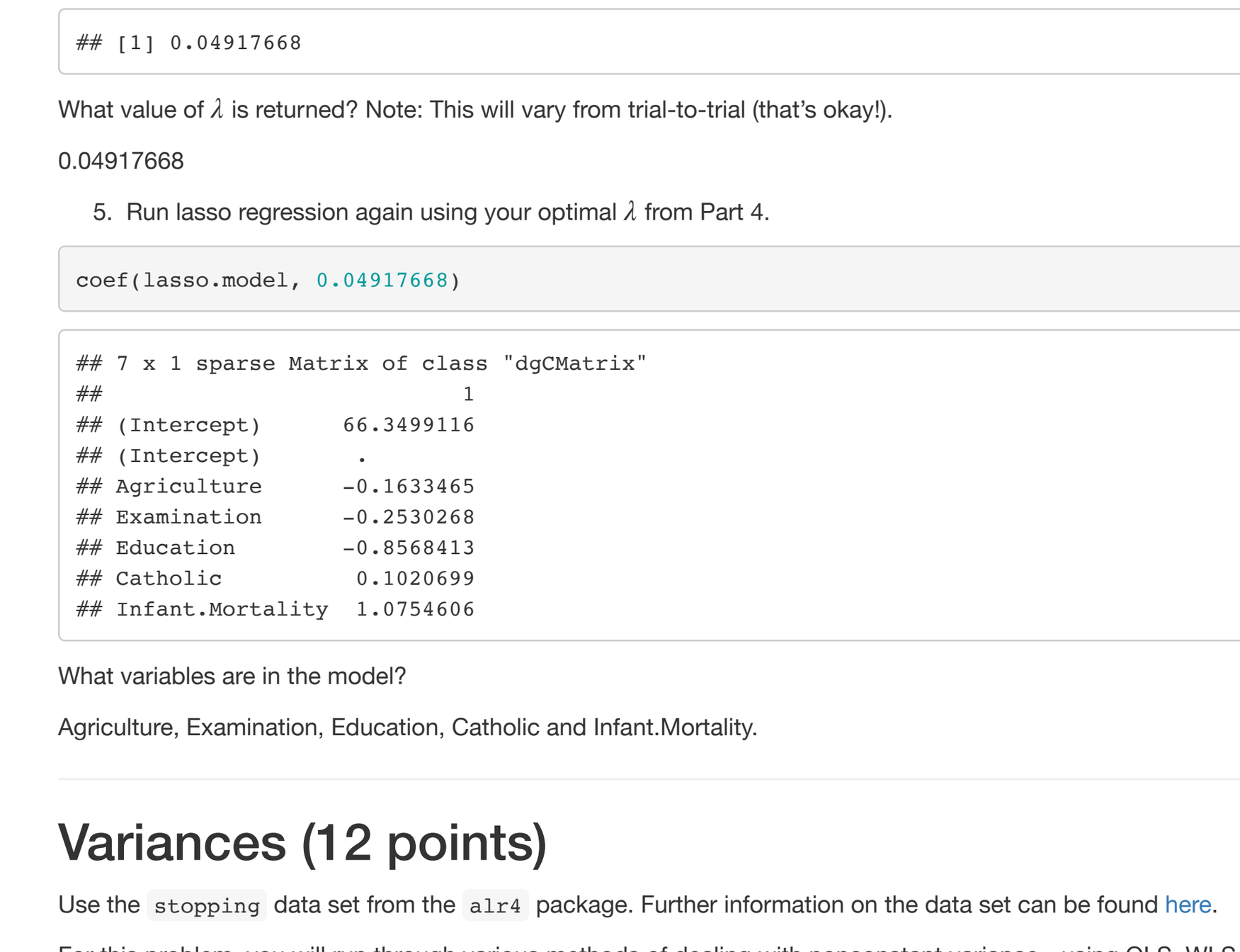
Use the `stopping` data set from the `alr4` package. Further information on the data set can be found here.

For this problem, you will run through various methods of dealing with nonconstant variance - using OLS, WLS (assuming known weights), WLS (with a sandwich estimator), and bootstrap.

1. Create a scatterplot of `Distance` versus `Speed` (Y vs X).

```
data(stopping, package = "alr4")

plot(stopping$Speed, stopping$Distance,
     main = "Distance vs. Speed",
     xlab = "Speed",
     ylab = "Distance")
```



2. Breifly explain why this graph supports fitting a quadratic regression model.

This graph supports fitting a quadratic regression model because it appears to be curvilinear.

3. Fit a quadratic model with constant variance (OLS). Remember to use the `I()` function to inhibit the squared term.

```
lm.quad <- lm(Distance ~ Speed + I(stopping$Speed^2), data = stopping)
summary(lm.quad)
```

```
##
## Call:
## lm(formula = Distance ~ Speed + I(stopping$Speed^2), data = stopping)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.5192  -5.4527  -0.5519   3.8442  27.9373
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.58036    5.10266   0.310    0.758
## Speed                0.41607    0.55641   0.748    0.458
## I(stopping$Speed^2)  0.06556    0.01303   5.033 4.83e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.927 on 59 degrees of freedom
## Multiple R-squared:  0.9144, Adjusted R-squared:  0.9115
## F-statistic: 315.3 on 2 and 59 DF,  p-value: < 2.2e-16
```

4. What are the standard errors for the two terms?

The standard error for Speed is 0.55641, whereas the standard error for Speed^2 is 0.01303.

5. Refit the quadratic model, but use WLS with the assumption that $Var(Distance|Speed) = Speed * \sigma^2$.

```
WLS <- stopping$Speed*var(stopping$Speed)

lm.quad2 <- lm(stopping$Distance ~  stopping$Speed + I(stopping$Speed^2), weights = WLS)
summary(lm.quad2)
```

```
##
## Call:
## lm(formula = stopping$Distance ~ stopping$Speed + I(stopping$Speed^2),
##     weights = WLS)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -1356.86  -179.48   -22.65   157.51  1531.23
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          3.02306    8.45341   0.358    0.722
## stopping$Speed       0.26952    0.76556   0.352    0.726
## I(stopping$Speed^2)  0.06867    0.01591   4.317 6.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 517.9 on 59 degrees of freedom
## Multiple R-squared:  0.894,  Adjusted R-squared:  0.8904
## F-statistic: 248.8 on 2 and 59 DF,  p-value: < 2.2e-16
```

6. What are the standard errors for the two terms?

The standard error for Speed is 0.76556, and the standard error for Speed^2 is 0.01591.

7. Using the unweighted model (OLS from Part 3), compute the standard errors using a sandwich estimator to correct for nonconstant variance. Use the `vcovHC()` function within the `sandwich` package. See the Lecture Notes for details.

```
library(sandwich)
sqrt(vcovHC(lm.quad))
```

```
## Warning in sqrt(vcovHC(lm.quad)): NaNs produced
```

```
##                     (Intercept)     Speed I(stopping$Speed^2)
## (Intercept)            4.295827       NaN          0.26286001
## Speed                       NaN 0.6303172                 NaN
## I(stopping$Speed^2)    0.262860       NaN          0.01724798
```

8. What are the standard errors for the two terms?

The standard error for Speed is 0.6303172 and the standard error for Speed^2 is 0.26286001.

9. Use bootstrapping to estimate the standard errors for the unweighted model (OLS from Part 3). See the example in the Lecture Notes for details. Because we have two terms here, you will need to create two empty vectors (with different names) to store your results. Use 10,000 (or more) repetitions. This should be done **manually** (like the Lecture Notes) without the use of any bootstrap R functions.

```
reps <- 10000
bootstrap.dist <- vector(length = reps)
bootstrap.dist2 <- vector(length = reps)

for(i in 1:reps){
  bootstrap.resample <- stopping[sample(1:nrow(stopping), nrow(stopping), replace = TRUE),]
  bootstrap.dist[i] <- lm(Distance ~ Speed, data = bootstrap.resample)$coef[2]
}


for (i in 1:reps) {
  bootstrap.resample2 <- stopping[sample(1:nrow(stopping), nrow(stopping), replace = TRUE),]
  bootstrap.dist2[i] <- lm(Distance ~ Speed^2, data = bootstrap.resample2)$coef[2]
}

sd(bootstrap.dist)
```

```
## [1] 0.2015769
```

```
sd(bootstrap.dist2)
```

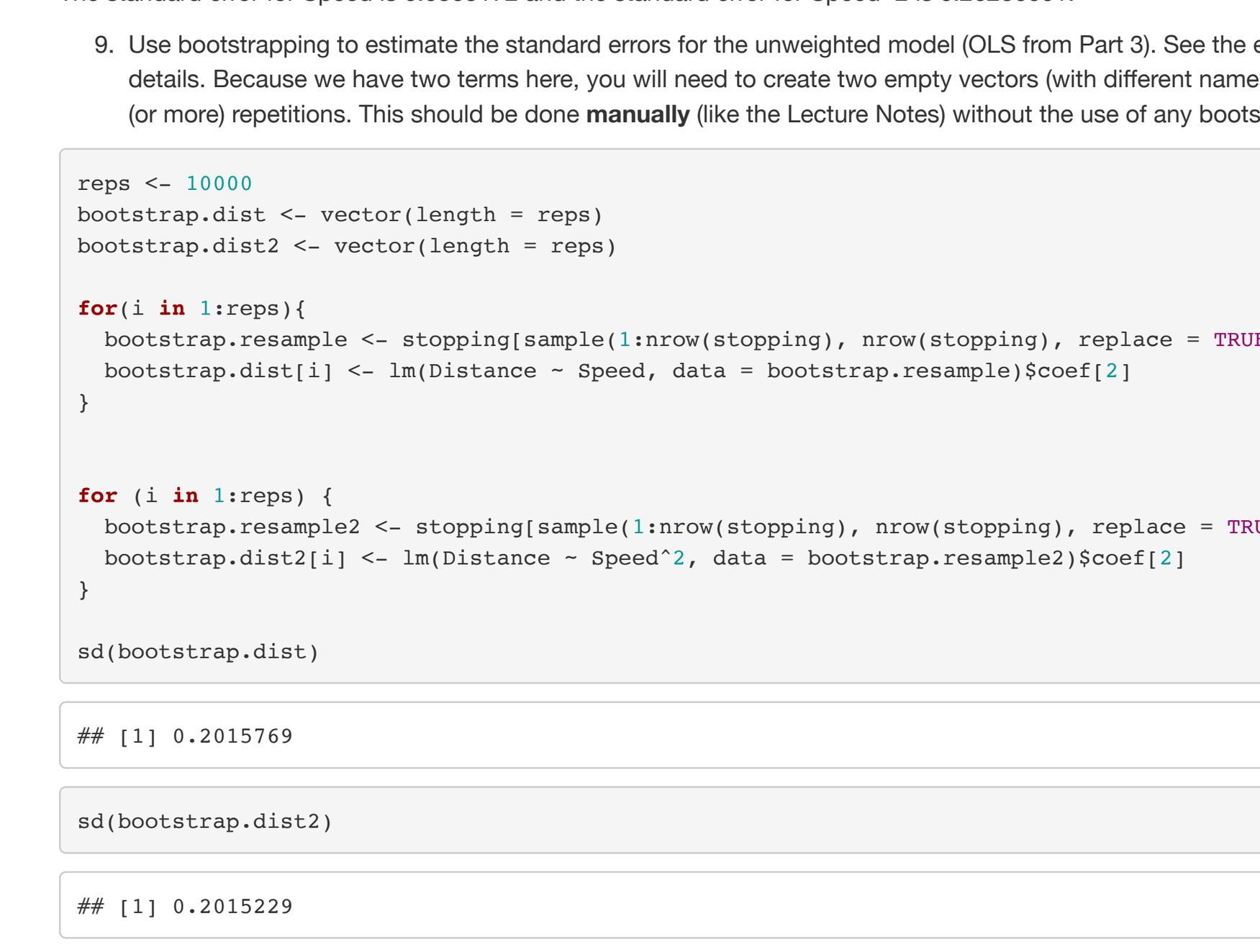
```
## [1] 0.2015229
```

10. What are the standard errors for the two terms?

sqrt(0.2010859) = 0.448426 is the standard error for Speed, and sqrt(0.2010052) = 0.448336 is the standard error for Speed^2.

11. Create 2 histograms (one for each term) to visualize the empirical distributions you created.

```
hist(bootstrap.dist,
     xlab = expression(hat(beta)),
     main = "Bootstrap Distribution 1")
```




```
hist(bootstrap.dist2,
     xlab = expression(hat(beta)),
     main = "Bootstrap Distribution 2")
```