Unit 4 - State Data Revisited

```r
data(state)
statedata = data.frame(state.x77)
str(statedata)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life.Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS.Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

Problem 1.1

```r
model1 <- lm(Life.Exp ~ ., statedata)
summary(model1)
```

```
##
## Call:
## lm(formula = Life.Exp ~ ., data = statedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48895 -0.51232 -0.02747  0.57002  1.49447
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
## Population   5.180e-05  2.919e-05   1.775   0.0832 .
## Income      -2.180e-05  2.444e-04  -0.089   0.9293
## Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
## Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
## HS.Grad      4.893e-02  2.332e-02   2.098   0.0420 *
## Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
## Area        -7.383e-08  1.668e-06  -0.044   0.9649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7448 on 42 degrees of freedom
## Multiple R-squared:  0.7362, Adjusted R-squared:  0.6922
## F-statistic: 16.74 on 7 and 42 DF,  p-value: 2.534e-10
```

0.6922

Problem 1.2

```r
pred <- predict(model1)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 23.29714
```

Problem 1.3

```r
model2 <- lm(Life.Exp ~ Population + Murder + Frost + HS.Grad, statedata)
summary(model2)
```

```
##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + Frost + HS.Grad,
##     data = statedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47095 -0.53464 -0.03701  0.57621  1.50683
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## Population   5.014e-05  2.512e-05   1.996  0.05201 .
## Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
## HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12
```

0.7126

Problem 1.4

```r
sum(model2$residuals^2)
```

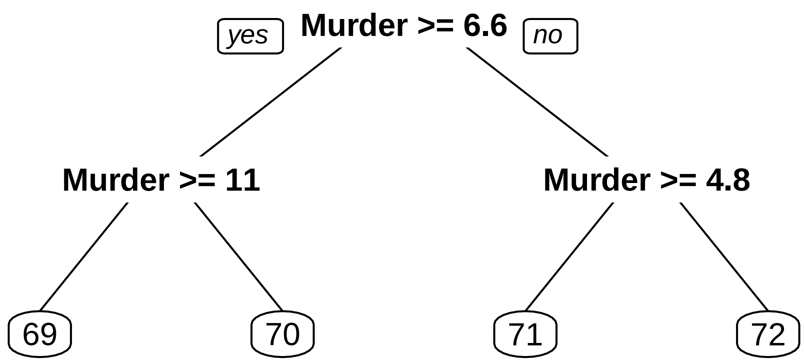
```
## [1] 23.30804
```

Problem 1.5

Trying different combinations of variables in linear regression is like trying different numbers of splits in a tree - this controls the complexity of the model.

Problem 2.1

```r
library(rpart)
library(rpart.plot)
CART_model1 <- rpart(Life.Exp ~ ., statedata)
prp(CART_model1)
```



Murder

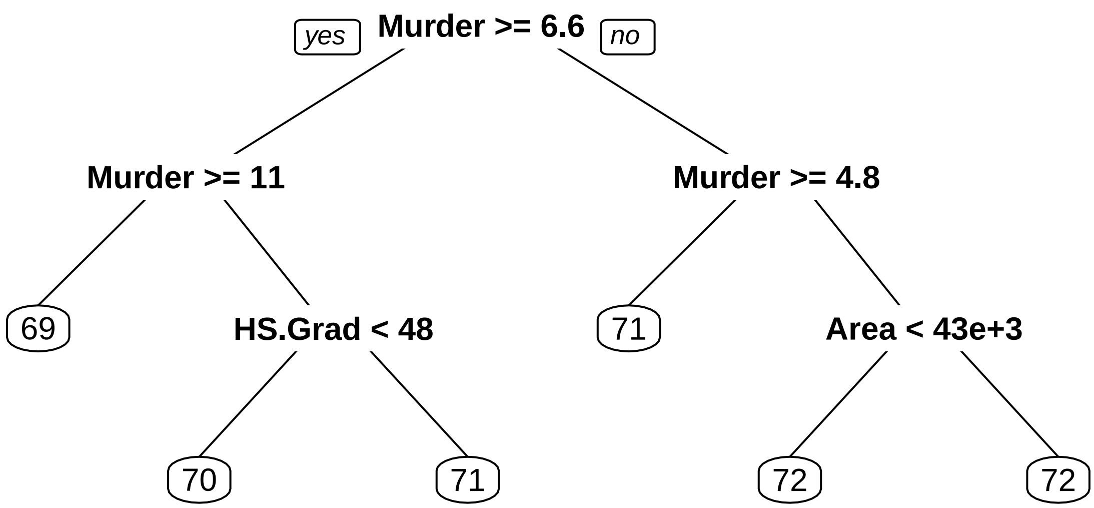
Problem 2.2

```r
pred <- predict(CART_model1)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 28.99848
```

Problem 2.3

```r
CART_model2 <- rpart(Life.Exp ~ ., statedata, minbucket = 5)
prp(CART_model2)
```



Murder, HS.Grad, Area

Problem 2.4

Larger
Since the tree now has more splits, it must be true that the default minbucket parameter was limiting the tree from splitting more before. So the default minbucket parameter must be larger than 5.

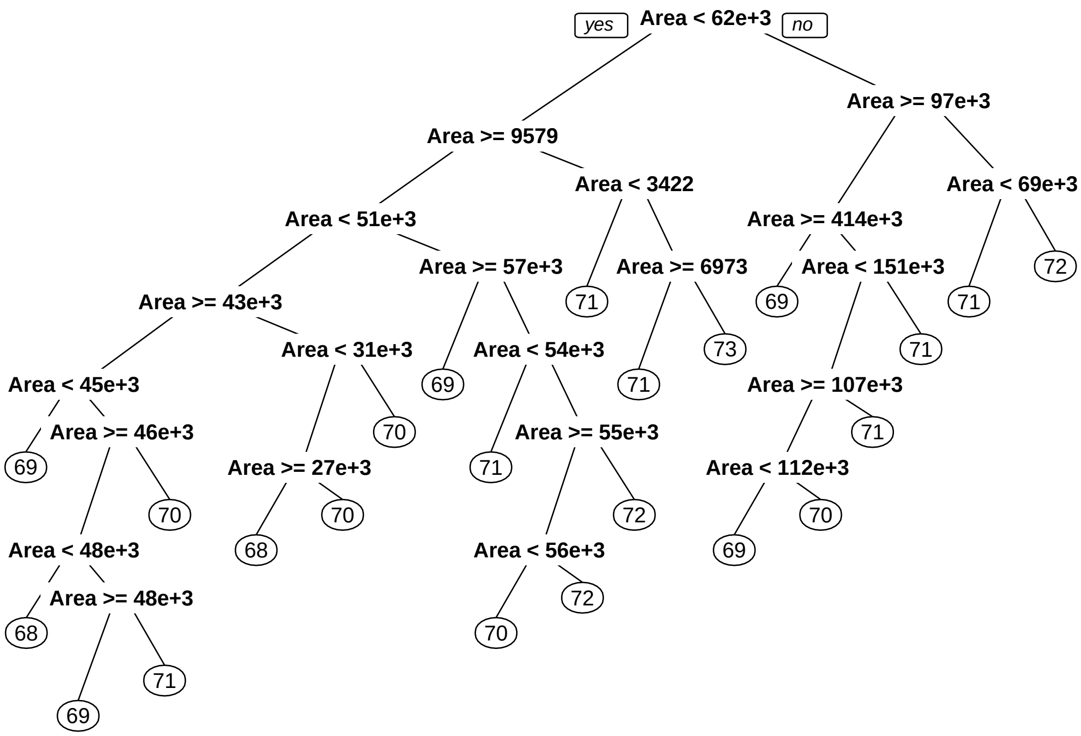
Problem 2.5

```r
pred <- predict(CART_model2)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 23.64283
```

Problem 2.6

```r
CART_model3 <- rpart(Life.Exp ~ Area, statedata, minbucket = 1)
prp(CART_model3)
```




```r
pred <- predict(CART_model3)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 9.312442
```

Problem 2.7

We can build almost perfect models given the right parameters, even if they violate our intuition of what a good model should be.
The correct answer is the second one. By making the minbucket parameter very small, we could build an almost perfect model using just one variable, that is not even our most significant variable. However, if you plot the tree using prp(CARTmodel3), you can see that the tree has 22 splits! This is not a very interpretable model, and will not generalize well.
The first answer is incorrect because our tree model that was not overfit performed similarly to the linear regression model. Trees only look better than linear regression here because we are overfitting the model to the data.
The third answer is incorrect because Area is not actually a very meaningful predictor. Without overfitting the tree, our model would not be very accurate only using Area.

Problem 3.1

```r
library(caret)
set.seed(111)
numFolds <- trainControl(method = "cv", number = 10)
cpGrid <- expand.grid(.cp = seq(0.01, 0.5, 0.01))
train(Life.Exp ~ ., statedata, method = "rpart", trControl = numFolds, tuneGrid = cpGrid)
```

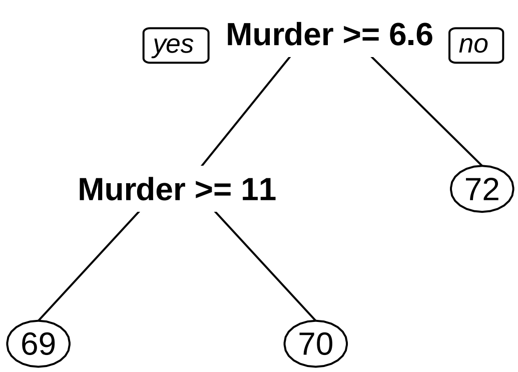
```
## CART
##
## 50 samples
##  7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 44, 45, 45, 46, 44, 45, ...
## Resampling results across tuning parameters:
##
##   cp    RMSE      Rsquared   MAE
##   0.01  1.042909  0.5206939  0.8299237
##   0.02  1.042909  0.5206939  0.8299237
##   0.03  1.027748  0.5338091  0.8173654
##   0.04  1.032567  0.5338091  0.8173654
##   0.05  1.032567  0.5338091  0.8173654
##   0.06  1.033866  0.5206565  0.8320776
##   0.07  1.029830  0.5285854  0.8252412
##   0.08  1.029830  0.5285854  0.8252412
##   0.09  1.029830  0.5285854  0.8252412
##   0.10  1.005814  0.5512315  0.8079269
##   0.11  1.005814  0.5512315  0.8079269
##   0.12  1.005814  0.5512315  0.8079269
##   0.13  1.032234  0.5238042  0.8262453
##   0.14  1.083214  0.5041955  0.8725504
##   0.15  1.106834  0.4822947  0.9050228
##   0.16  1.138118  0.4775423  0.9422217
##   0.17  1.174001  0.4287787  0.9676503
##   0.18  1.192122  0.3990629  0.9942598
##   0.19  1.192122  0.3990629  0.9942598
##   0.20  1.192122  0.3990629  0.9942598
##   0.21  1.192122  0.3990629  0.9942598
##   0.22  1.192122  0.3990629  0.9942598
##   0.23  1.192122  0.3990629  0.9942598
##   0.24  1.192122  0.3990629  0.9942598
##   0.25  1.192122  0.3990629  0.9942598
##   0.26  1.192122  0.3990629  0.9942598
##   0.27  1.192122  0.3990629  0.9942598
##   0.28  1.192122  0.3990629  0.9942598
##   0.29  1.192122  0.3990629  0.9942598
##   0.30  1.192122  0.3990629  0.9942598
##   0.31  1.192122  0.3990629  0.9942598
##   0.32  1.192122  0.3990629  0.9942598
##   0.33  1.192122  0.3990629  0.9942598
##   0.34  1.192122  0.3990629  0.9942598
##   0.35  1.192122  0.3990629  0.9942598
##   0.36  1.192122  0.3990629  0.9942598
```

```
##   0.37  1.192122  0.3990629  0.9942598
##   0.38  1.192122  0.3990629  0.9942598
##   0.39  1.192122  0.3990629  0.9942598
##   0.40  1.192122  0.3990629  0.9942598
##   0.41  1.192122  0.3990629  0.9942598
##   0.42  1.192122  0.3990629  0.9942598
##   0.43  1.192122  0.3990629  0.9942598
##   0.44  1.192122  0.3990629  0.9942598
##   0.45  1.192122  0.3990629  0.9942598
##   0.46  1.308759  0.3091923  1.0963695
##   0.47  1.309534  0.3328614  1.0922317
##   0.48  1.358580  0.2908310  1.1126138
##   0.49  1.335777  0.3552299  1.0771938
##   0.50  1.361946  0.2358921  1.1229974
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.12.
```

0.12

Problem 3.2

```
CART_model4 <- rpart(Life.Exp ~ ., statedata, cp = 0.12)
prp(CART_model4)
```



6.6, 11

Problem 3.3

```r
pred <- predict(CART_model4)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 32.86549
```

Problem 3.4

The model we just made with the “best” cp
The purpose of cross-validation is to pick the tree that will perform the best on a test set. So we would expect the model we made with the “best” cp to perform best on a test set.

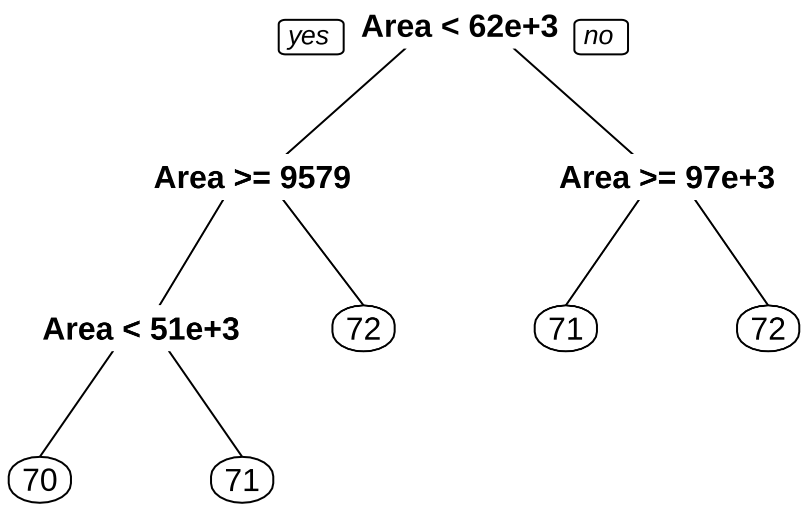
Problem 3.5

```r
set.seed(111)
train(Life.Exp ~ Area, statedata, method = "rpart",
      trControl = numFolds, tuneGrid = cpGrid)
```

```
## CART
##
## 50 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 44, 45, 45, 46, 44, 45, ...
## Resampling results across tuning parameters:
##
##   cp    RMSE      Rsquared     MAE
##   0.01  1.285085  0.242889993  1.068979
##   0.02  1.297420  0.227958081  1.077964
##   0.03  1.297420  0.227958081  1.077964
##   0.04  1.297420  0.227958081  1.077964
##   0.05  1.297420  0.227958081  1.077964
##   0.06  1.283241  0.255224868  1.071996
##   0.07  1.283241  0.255224868  1.071996
##   0.08  1.277535  0.253025061  1.054684
##   0.09  1.286127  0.239816630  1.060619
##   0.10  1.286127  0.239816630  1.060619
##   0.11  1.286127  0.239816630  1.060619
##   0.12  1.278550  0.239816630  1.060619
##   0.13  1.336117  0.205007172  1.116064
##   0.14  1.364618  0.132092640  1.125916
##   0.15  1.364016  0.272311296  1.124202
```

```
##   0.16  1.348422  0.216040174  1.128918
##   0.17  1.365452  0.125811897  1.110581
##   0.18  1.334937  0.006222148  1.103295
##   0.19  1.328891          NaN  1.099663
##   0.20  1.328891          NaN  1.099663
##   0.21  1.328891          NaN  1.099663
##   0.22  1.328891          NaN  1.099663
##   0.23  1.328891          NaN  1.099663
##   0.24  1.328891          NaN  1.099663
##   0.25  1.328891          NaN  1.099663
##   0.26  1.328891          NaN  1.099663
##   0.27  1.328891          NaN  1.099663
##   0.28  1.328891          NaN  1.099663
##   0.29  1.328891          NaN  1.099663
##   0.30  1.328891          NaN  1.099663
##   0.31  1.328891          NaN  1.099663
##   0.32  1.328891          NaN  1.099663
##   0.33  1.328891          NaN  1.099663
##   0.34  1.328891          NaN  1.099663
##   0.35  1.328891          NaN  1.099663
##   0.36  1.328891          NaN  1.099663
##   0.37  1.328891          NaN  1.099663
##   0.38  1.328891          NaN  1.099663
##   0.39  1.328891          NaN  1.099663
##   0.40  1.328891          NaN  1.099663
##   0.41  1.328891          NaN  1.099663
##   0.42  1.328891          NaN  1.099663
##   0.43  1.328891          NaN  1.099663
##   0.44  1.328891          NaN  1.099663
##   0.45  1.328891          NaN  1.099663
##   0.46  1.328891          NaN  1.099663
##   0.47  1.328891          NaN  1.099663
##   0.48  1.328891          NaN  1.099663
##   0.49  1.328891          NaN  1.099663
##   0.50  1.328891          NaN  1.099663
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.08.
```

```r
CART_model5 <- rpart(Life.Exp ~ Area, statedata, cp = 0.01)
prp(CART_model5)
```



4 splits

Problem 3.6

9579, 51000

Problem 3.7

```
pred <- predict(CART_model5)
SSE <- sum((pred - statedata$Life.Exp)^2)
SSE
```

```
## [1] 44.26817
```

The Area variable is not as predictive as Murder rate.
The original Area tree was overfitting the data - it was uninterpretable. Area is not as useful as Murder - if it was, it would have been in the cross-validated tree. Cross-validation is not designed to improve the fit on the training data, but it won't necessarily make it worse either.
Cross-validation cannot guarantee improving the SSE on unseen data, although it often helps.