Assaf Bitton

You can run the following code to prepare the analysis.

```
library(r02pro)     #INSTALL IF NECESSARY

## Warning: package 'r02pro' was built under R version 4.4.3

library(tidyverse)  #INSTALL IF NECESSARY

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'tidyr' was built under R version 4.4.2

## Warning: package 'dplyr' was built under R version 4.4.2

library(MASS)

## Warning: package 'MASS' was built under R version 4.4.3

my_ahp <- ahp %>%
  dplyr::select(gar_car, liv_area, oa_qual, sale_price) %>%
  na.omit() %>%
  mutate(type = factor(ifelse(sale_price > median(sale_price), "Expensive", "
Cheap")))
tr_ind <- 1:(nrow(my_ahp)/20)
my_ahp_train <- my_ahp[tr_ind, ]
my_ahp_test <- my_ahp[-tr_ind, ]
```

Suppose we want to build a tree to predict `sale_price` and `type` using `gar_car`, `liv_area` and `oa_qual`. Please answer the following questions.

Q1

First, we fit a deep regression tree to predict `sale_price` using the training data `my_ahp_train`. Note that, here we use `tree.control` to generate such a deep tree, which is then stored into object `tree_fit`.

```
library(tree)

## Warning: package 'tree' was built under R version 4.4.3

n <- nrow(my_ahp_train)
my_control <- tree.control(nobs = n, minsize = 2, mindev = 0)
tree_fit <- tree(sale_price ~ gar_car + liv_area + oa_qual,
                 control = my_control,
                 data = my_ahp_train)
```

In the following questions (a)–(c), we will find the optimal subtree of above deep tree, `tree_fit`, via cost complexity pruning. *Note: Use `set.seed(0)` before calling `cv.tree` to ensure reproducibility.*

(a) Create a plot illustrating the relation between the cross-validation error and the tree size.

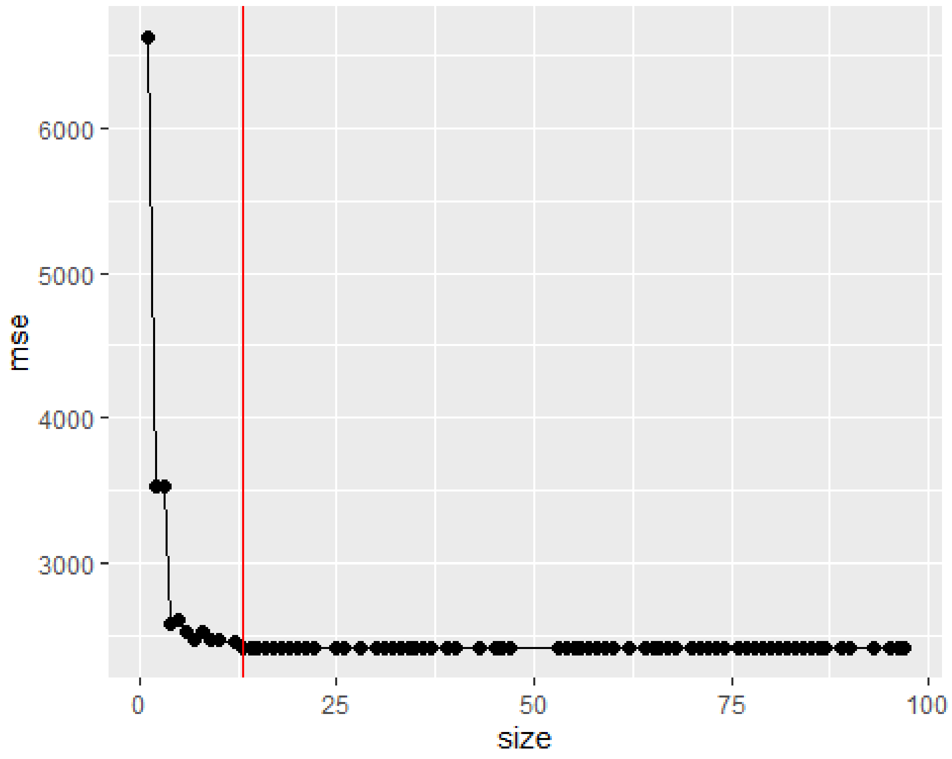
Solution:

```r
set.seed(0)

cv_sal_tree <- cv.tree(tree_fit)

best_size <- with(cv_sal_tree, min(size[dev == min(dev)]))

ggplot(data.frame(size = cv_sal_tree$size,
                  mse = cv_sal_tree$dev/n),
       mapping = aes(x = size, y = mse)) +
  geom_point(size = 2) +
  geom_line() +
  geom_vline(xintercept = best_size, col = "red")
```

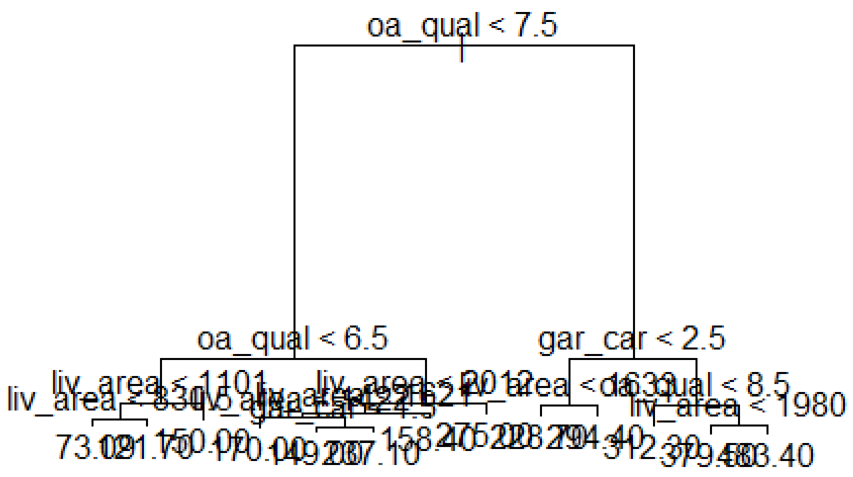


(b) Generate a pruned tree with the number of leaf nodes determined by cross-validation and visualize the tree. Note that if many trees have the same cross-validation error, you want to choose the tree with the fewest leaf nodes.

Solution:

```r
sal_tree_final <- prune.tree(tree_fit, best = best_size)
```

```r
plot(sal_tree_final)
text(sal_tree_final)
```



(c) Use the pruned tree in part (b) to calculate the training MSE and test MSE.

Solution:

```r
pred_tr <- predict(sal_tree_final,
                    newdata = my_ahp_train)
pred_te <- predict(sal_tree_final,
                   newdata = my_ahp_test)

mse_tr <- mean((pred_tr - my_ahp_train$sale_price)^2)
mse_te <- mean((pred_te - my_ahp_test$sale_price)^2)

mse_tr

## [1] 644.8815

mse_te

## [1] 2040.102
```

Q2

Build a classification tree with the number of leaf nodes determined via cross-validation to predict type using `gar_car`, `liv_area` and `oa_qual` with the training data `my_ahp_train`.

Visualize the tree and provide interpretations. *Note: Use the cross-entropy criterion for tree-growing and the classification error criterion for pruning; Use `set.seed(0)` before calling `cv.tree` to ensure reproducibility.*

Solution:

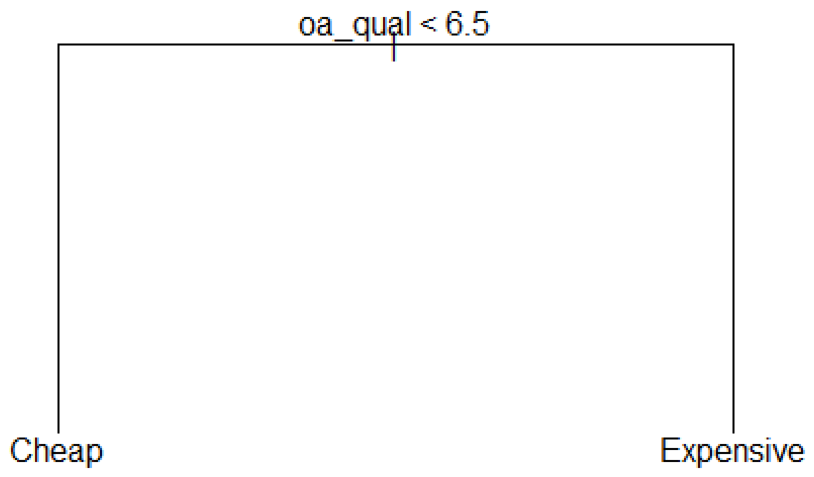
```r
sal_tree <- tree(type ~ gar_car +
                   liv_area +
                   oa_qual,
                 data = my_ahp_train)

set.seed(0)
cv_sal_tree <- cv.tree(sal_tree,
                       method = "misclass")

best_size <- with(cv_sal_tree,
                  min(size[dev == min(dev)]))

sal_tree_final <- prune.tree(sal_tree,
                             best = best_size,
                             method = "misclass")

plot(sal_tree_final)
text(sal_tree_final)
```



- The result of this *tree* is due to the fact that **oa_qual** is the most *significant predictor*. The cross validation process, using the miss classifications error criteria, determined that a tree with a **single split** provided the best balance between accuracy and model simplicity. - The model uses a **single** threshold of **6.5** for **oa_qual** to split the data. **Cheap** classification is when a house has an *overall quality* less than *6.5*. **Expensive** is when a house has an *overall quality* greater than *6.5*. - Because the tree was pruned to its smallest **optimal size**, adding variables **liv_area** and **gar_car** were found to not significantly improve classification accuracy beyond what the first split provided. - Using **min(size[dev == min(dev)])** ensured that the simplest possible model was chosen among those that achieved the lowest error rate, preventing over fitting to the training data.