Problem 2

2025-09-24

Problem 2:

```
# NOTE: Data files must be in a 'Data' subdirectory relative to this R Markdown file
# Expected structure:
#   - prob4.Rmd
#   - Data/
#     |- laptop_train.csv
#     |- laptop_test.csv
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# Read CSV files using relative paths
df_orig <- read_csv("insurance_charges.csv")
```

```
## Rows: 1338 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): age, bmi, charges, f_bmi, cardiovascular_care_cost
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question (a)

```
head(df_orig)
```

```
## # A tibble: 6 x 5
##     age   bmi charges f_bmi cardiovascular_care_cost
##   <dbl> <dbl>   <dbl> <dbl>                    <dbl>
## 1    19  3.90  16885. 0.591                    1876.
## 2    18  5.19   1726. 0.716                    2466.
## 3    28  5.00   4449. 0.699                    2473.
## 4    33  3.03  21984. 0.481                    1656.
## 5    32  4.09   3867. 0.612                    1933.
## 6    31  3.51   3757. 0.545                    1548.
```

```
# We want to know the dimensions of our dataset.
dim(df_orig) # there are 9 features and 665 observations
```

```
## [1] 1338    5
```

```r
str(df_orig) # structure (variable types, first few entries)
```

```
## spc_tbl_ [1,338 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age                     : num [1:1338] 19 18 28 33 32 31 46 37 37 60 ...
##  $ bmi                     : num [1:1338] 3.9 5.19 5 3.03 4.09 ...
##  $ charges                 : num [1:1338] 16885 1726 4449 21984 3867 ...
##  $ f_bmi                   : num [1:1338] 0.591 0.716 0.699 0.481 0.612 ...
##  $ cardiovascular_care_cost: num [1:1338] 1876 2466 2473 1656 1933 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   bmi = col_double(),
##   ..   charges = col_double(),
##   ..   f_bmi = col_double(),
##   ..   cardiovascular_care_cost = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
summary(df_orig) # summary statistics by column
```

```
##       age             bmi             charges           f_bmi
##  Min.   :18.00   Min.   : 2.179   Min.   : 1122   Min.   :0.3382
##  1st Qu.:27.00   1st Qu.: 3.607   1st Qu.: 4740   1st Qu.:0.5572
##  Median :39.00   Median : 4.407   Median : 9382   Median :0.6442
##  Mean   :39.21   Mean   : 4.672   Mean   :13270   Mean   :0.6497
##  3rd Qu.:51.00   3rd Qu.: 5.434   3rd Qu.:16640   3rd Qu.:0.7351
##  Max.   :64.00   Max.   :13.359   Max.   :63770   Max.   :1.1258
##  cardiovascular_care_cost
##  Min.   : 827.1
##  1st Qu.:1784.1
##  Median :2204.5
##  Mean   :2338.0
##  3rd Qu.:2720.7
##  Max.   :6621.8
```

```r
# Split the data into training and test (70% train, 30% test)
set.seed(15072)
# training (70%) and test (30%) partition
smp_size <- floor(0.70 * nrow(df_orig))
train_ind <- sample(seq_len(nrow(df_orig)), size = smp_size, replace = FALSE)
df_train <- df_orig[train_ind, ]
df_test <- df_orig[-train_ind, ]

# Check the dimensions of the training and test sets
dim(df_train)
```

```
## [1] 936   5
```

```r
dim(df_test)
```

```
## [1] 402   5
```

```r
# Use glimpse to get a quick overview of both datasets
glimpse(df_train)
```

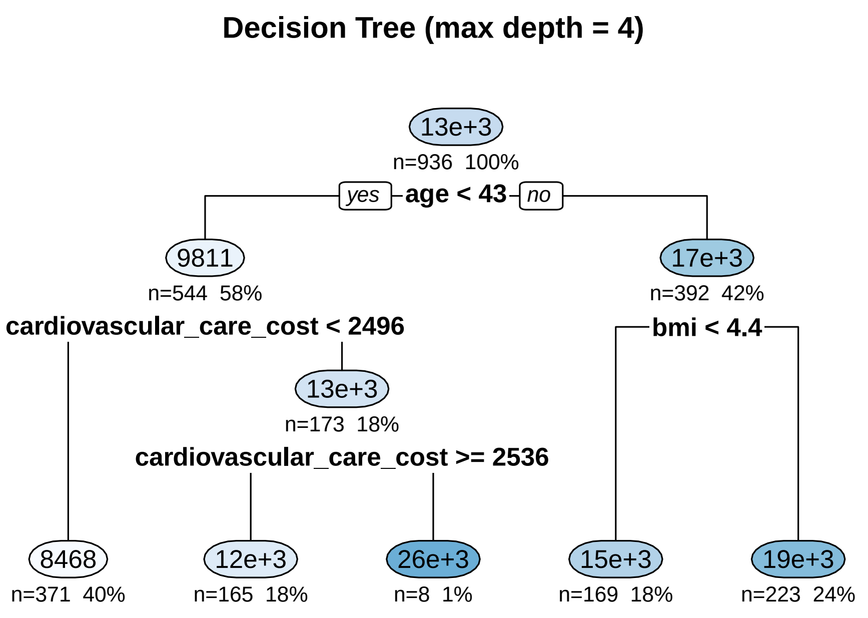
```
## Rows: 936
## Columns: 5
## $ age                      <dbl> 63, 19, 44, 21, 34, 40, 56, 38, 47, 34, 36, 3~
## $ bmi                      <dbl> 5.877215, 3.306357, 3.825654, 3.504007, 3.825~
## $ charges                  <dbl> 13887.204, 2709.112, 7626.993, 17942.106, 500~
## $ f_bmi                    <dbl> 0.7691716, 0.5193497, 0.5827057, 0.5445650, 0~
## $ cardiovascular_care_cost <dbl> 2949.277, 1530.945, 1821.328, 1364.722, 2143.~
```

```r
glimpse(df_test)
```

```
## Rows: 402
## Columns: 5
## $ age                      <dbl> 33, 32, 31, 37, 62, 56, 52, 23, 59, 22, 28, 3~
## $ bmi                      <dbl> 3.027632, 4.092107, 3.510852, 4.286243, 3.606~
## $ charges                  <dbl> 21984.471, 3866.855, 3756.622, 6406.411, 2780~
## $ f_bmi                    <dbl> 0.4811030, 0.6119470, 0.5454126, 0.6320768, 0~
## $ cardiovascular_care_cost <dbl> 1656.104, 1933.298, 1548.085, 1752.809, 1625.~
```

Question (b)

```r
# Fit a decision tree model with no depth greater than 4
tree_model <- rpart(charges ~ ., data = df_train, control = rpart.control(maxdepth = 4))


# Plot the decision tree
rpart.plot(tree_model, type = 2, extra = 101, under = TRUE,
           main = "Decision Tree (max depth = 4)")
```



```
# Summary of the model
summary(tree_model)
```

```
## Call:
## rpart(formula = charges ~ ., data = df_train, control = rpart.control(maxdepth = 4))
##   n= 936
##
##           CP nsplit rel error    xerror       xstd
## 1 0.09583610      0 1.0000000 1.0041683 0.06325607
## 2 0.01611526      1 0.9041639 0.9188786 0.05829077
## 3 0.01181389      2 0.8880486 0.9629522 0.05839692
## 4 0.01102792      3 0.8762347 0.9791362 0.05889894
## 5 0.01000000      4 0.8652068 0.9902592 0.05904446
##
## Variable importance
##                      age cardiovascular_care_cost                      bmi
##                       50                       21                       14
##                    f_bmi
##                       14
##
## Node number 1: 936 observations,    complexity param=0.0958361
##   mean=12914.59, MSE=1.395087e+08
##   left son=2 (544 obs) right son=3 (392 obs)
##   Primary splits:
##       age                      < 42.5      to the left,  improve=0.09583610, (0 missing)
##       bmi                      < 4.392632  to the left,  improve=0.03606740, (0 missing)
##       f_bmi                    < 0.6427244 to the left,  improve=0.03606740, (0 missing)
##       cardiovascular_care_cost < 2493.027  to the left,  improve=0.03167217, (0 missing)
##   Surrogate splits:
##       cardiovascular_care_cost < 3015.68   to the left,  agree=0.597, adj=0.038, (0 split)
##       bmi                      < 5.728587  to the left,  agree=0.593, adj=0.028, (0 split)
##       f_bmi                     < 0.7580472 to the left,  agree=0.593, adj=0.028, (0 split)
##
## Node number 2: 544 observations,    complexity param=0.01611526
##   mean=9810.683, MSE=1.159912e+08
##   left son=4 (371 obs) right son=5 (173 obs)
##   Primary splits:
##       cardiovascular_care_cost < 2496.317  to the left,  improve=0.03334962, (0 missing)
##       bmi                      < 4.357944  to the left,  improve=0.02996210, (0 missing)
##       f_bmi                    < 0.6392812 to the left,  improve=0.02996210, (0 missing)
##       age                      < 28.5      to the left,  improve=0.01657340, (0 missing)
##   Surrogate splits:
##       bmi   < 4.991013  to the left,  agree=0.932, adj=0.786, (0 split)
##       f_bmi < 0.6981874 to the left,  agree=0.932, adj=0.786, (0 split)
##
## Node number 3: 392 observations,    complexity param=0.01181389
##   mean=17222.06, MSE=1.402212e+08
##   left son=6 (169 obs) right son=7 (223 obs)
##   Primary splits:
##       bmi                      < 4.392097  to the left,  improve=0.02806535, (0 missing)
##       f_bmi                    < 0.6426714 to the left,  improve=0.02806535, (0 missing)
##       age                      < 58.5      to the left,  improve=0.02175027, (0 missing)
##       cardiovascular_care_cost < 2163.841  to the left,  improve=0.02065833, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.6426714 to the left,  agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 2200.411  to the left,  agree=0.926, adj=0.828, (0 split)
##
## Node number 4: 371 observations
##   mean=8467.627, MSE=7.249573e+07
##
## Node number 5: 173 observations,    complexity param=0.01102792
##   mean=12690.88, MSE=1.971037e+08
##   left son=10 (165 obs) right son=11 (8 obs)
##   Primary splits:
##       cardiovascular_care_cost < 2535.65   to the right, improve=0.04223086, (0 missing)
##       bmi                      < 8.080433  to the right, improve=0.02730380, (0 missing)
##       f_bmi                    < 0.9074317 to the right, improve=0.02730380, (0 missing)
##       age                      < 18.5      to the left,  improve=0.01676663, (0 missing)
##
## Node number 6: 169 observations
##   mean=14943.28, MSE=5.366741e+07
##
## Node number 7: 223 observations
##   mean=18949.02, MSE=1.988979e+08
##
## Node number 10: 165 observations
##   mean=12055.6, MSE=1.872335e+08
##
## Node number 11: 8 observations
##   mean=25793.53, MSE=2.206731e+08
```

```r
# Predictions on training and test sets
test_pred <- predict(tree_model, newdata = df_test)

# Calculate R-squared for training and test sets
```

```r
y_true <- df_test$charges
sse <- sum((y_true - test_pred)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_basetree <- 1 - sse/sst

print(paste("R-squared on test set:", round(R2_basetree, 4)))
```

```
## [1] "R-squared on test set: 0.0557"
```

Question (c)

Residuals

The residuals are the differences between the actual charges in the test dataset and the predicted charges from the basetree model:

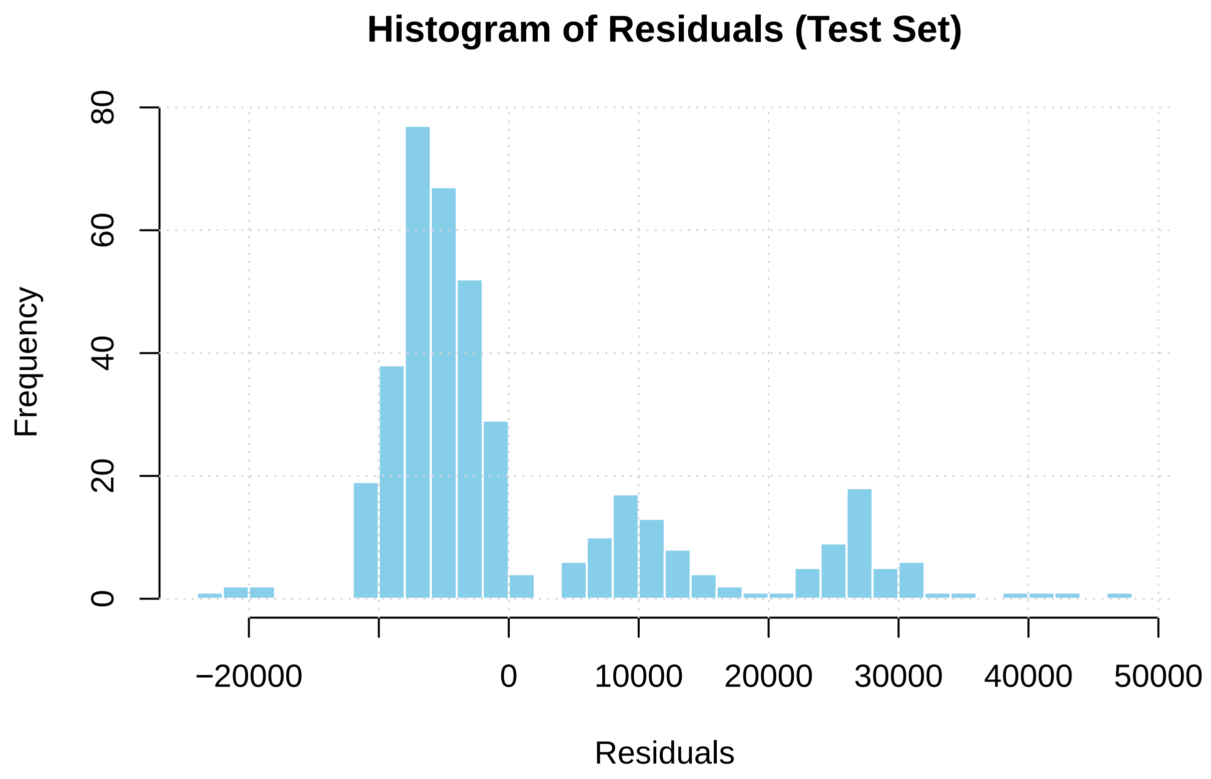
$$\text{residual}_i = y_i - \hat{y}_i$$

where
- y_i = actual charge for observation i (from the test dataset),
- $\hat{y}_i$ = predicted charge for observation i (from the basetree).

```r
# Calculate residuals on the test set
residuals <- df_test$charges - test_pred

# Summarize the residuals
residual_summary <- summary(residuals)
print(residual_summary)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -23532.0  -6731.6  -4071.9    819.6   7268.4  46515.5
```

```r
# Plot a histogram of the residuals
hist(residuals, breaks = 40, main = "Histogram of Residuals (Test Set)",
     xlab = "Residuals", col = "skyblue", border = "white")
grid()
```



Question (d)

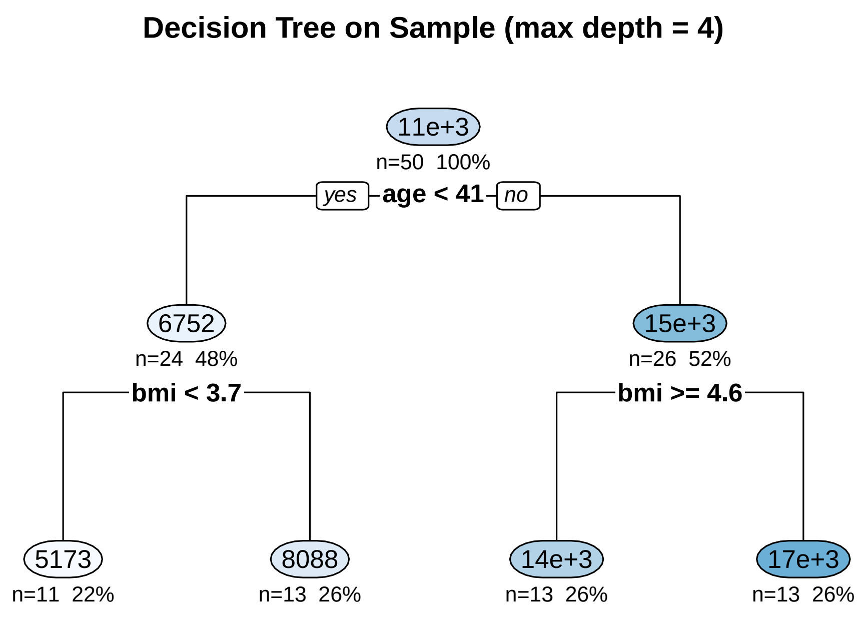
1. First we take a random sample with replacement of size 50 from the training set.

```
# Take random sample with replacement of size 50 from the training set
df_sample <- df_train[sample(nrow(df_train), 50, replace = TRUE), ]

# Check the dimensions of the sample
dim(df_sample)
```

```
## [1] 50  5
```

2. Next, we fit a decision tree model to this sample, again with max depth 4.

```
# Fit a decision tree model to the sample
tree_model_sample <- rpart(charges ~ ., data = df_sample,
                           control = rpart.control(maxdepth = 4))

# Plot the decision tree
rpart.plot(tree_model_sample, type = 2, extra = 101, under = TRUE,
           main = "Decision Tree on Sample (max depth = 4)")
```



```r
# Summary of the model
summary(tree_model_sample)
```

```
## Call:
## rpart(formula = charges ~ ., data = df_sample, control = rpart.control(maxdepth = 4))
##   n= 50
##
##           CP nsplit rel error   xerror      xstd
## 1 0.18784602      0 1.0000000 1.023816 0.3615694
## 2 0.01428744      1 0.8121540 0.980413 0.3669908
## 3 0.01060455      2 0.7978665 1.133964 0.3717291
## 4 0.01000000      3 0.7872620 1.144902 0.3722554
##
## Variable importance
##                      age                      bmi                    f_bmi
##                       45                       20                       20
## cardiovascular_care_cost
##                       15
##
## Node number 1: 50 observations,    complexity param=0.187846
##   mean=11159.72, MSE=9.548714e+07
##   left son=2 (24 obs) right son=3 (26 obs)
##   Primary splits:
##       age                      < 40.5      to the left,  improve=0.18784600, (0 missing)
##       bmi                      < 4.223596  to the left,  improve=0.06009507, (0 missing)
##       f_bmi                    < 0.625667  to the left,  improve=0.06009507, (0 missing)
##       cardiovascular_care_cost < 2294.667  to the left,  improve=0.04458673, (0 missing)
##   Surrogate splits:
##       bmi                      < 3.879789  to the left,  agree=0.68, adj=0.333, (0 split)
##       f_bmi                    < 0.5887976 to the left,  agree=0.68, adj=0.333, (0 split)
##       cardiovascular_care_cost < 1984.881  to the left,  agree=0.64, adj=0.250, (0 split)
```

```
##
## Node number 2: 24 observations,    complexity param=0.01060455
##   mean=6751.585, MSE=5.709371e+07
##   left son=4 (11 obs) right son=5 (13 obs)
##   Primary splits:
##       bmi                      < 3.701539  to the left,  improve=0.03694943, (0 missing)
##       f_bmi                    < 0.5681381 to the left,  improve=0.03694943, (0 missing)
##       age                      < 28.5      to the right, improve=0.02311741, (0 missing)
##       cardiovascular_care_cost < 2262.969  to the left,  improve=0.02229621, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.5681381 to the left,  agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 1781.719  to the left,  agree=0.917, adj=0.818, (0 split)
##       age                      < 27        to the left,  agree=0.708, adj=0.364, (0 split)
##
## Node number 3: 26 observations,    complexity param=0.01428744
##   mean=15228.76, MSE=9.643323e+07
##   left son=6 (13 obs) right son=7 (13 obs)
##   Primary splits:
##       bmi                      < 4.621298  to the right, improve=0.02720629, (0 missing)
##       f_bmi                    < 0.6647613 to the right, improve=0.02720629, (0 missing)
##       age                      < 57.5      to the left,  improve=0.02423591, (0 missing)
##       cardiovascular_care_cost < 2294.667  to the left,  improve=0.02419122, (0 missing)
##   Surrogate splits:
##       f_bmi                    < 0.6647613 to the right, agree=1.000, adj=1.000, (0 split)
##       cardiovascular_care_cost < 2070.5    to the right, agree=0.846, adj=0.692, (0 split)
##       age                      < 47.5      to the right, agree=0.615, adj=0.231, (0 split)
##
## Node number 4: 11 observations
##   mean=5172.617, MSE=2.668401e+07
##
## Node number 5: 13 observations
##   mean=8087.635, MSE=7.893039e+07
##
## Node number 6: 13 observations
##   mean=13609.01, MSE=8.389341e+07
##
## Node number 7: 13 observations
##   mean=16848.51, MSE=1.037259e+08
```

3. Finally, we compute the R-squared on the test set using this new model.

```r
# Predictions on test set using the model fitted to the sample
test_pred_sample <- predict(tree_model_sample, newdata = df_test)

# Calculate R-squared for test set using the sample model
y_true <- df_test$charges
sse_sample <- sum((y_true - test_pred_sample)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_sample <- 1 - sse_sample/sst
print(paste("R-squared on test set (sample model):", round(R2_sample, 4)))
```

```
## [1] "R-squared on test set (sample model): -0.0404"
```

Question (e)

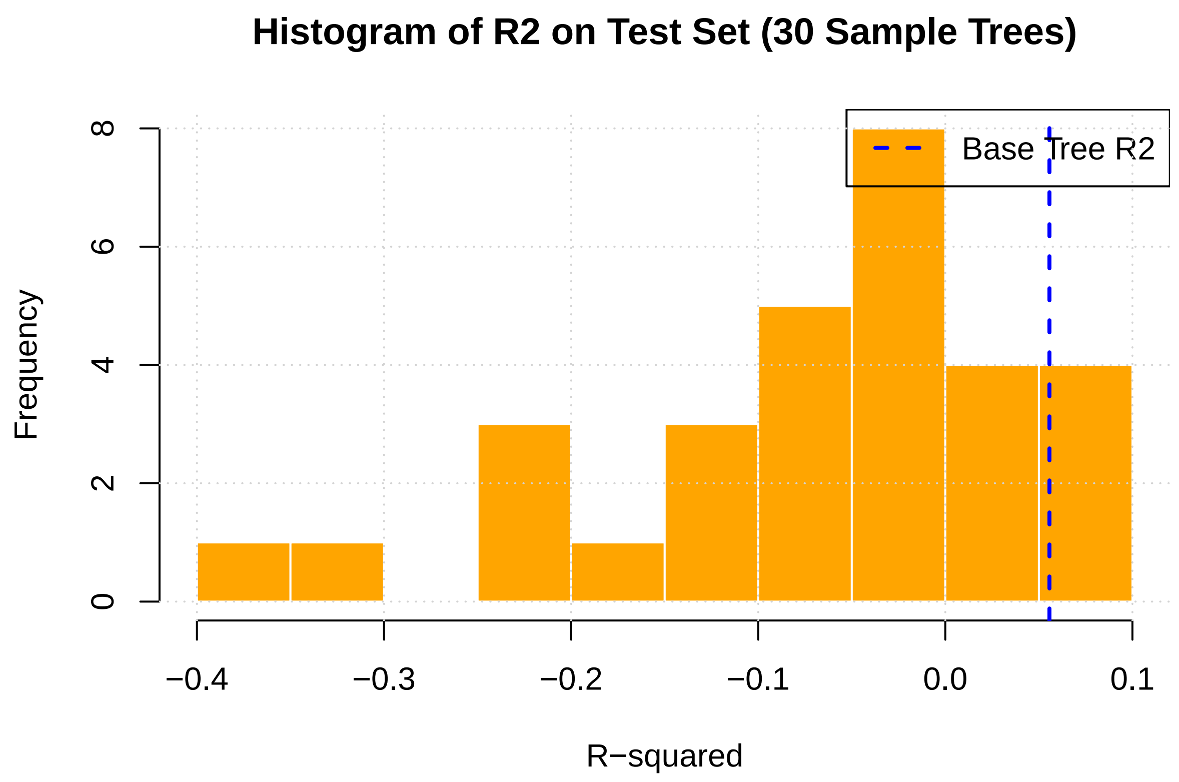
```r
# Repeat the sampling, training, and R2 calculation 30 times
R2_samples <- numeric(30)
wise_tree_models <- vector("list", 30) # Save each tree for use in (f)

for (i in 1:30) {
  # Sample with replacement
  df_sample <- df_train[sample(nrow(df_train), 50, replace = TRUE), ]
  # Fit tree
  tree_model_sample <- rpart(charges ~ ., data = df_sample,
                             control = rpart.control(maxdepth = 4))
  # Store the model
  wise_tree_models[[i]] <- tree_model_sample
  # Predict on test set
  test_pred_sample <- predict(tree_model_sample, newdata = df_test)
  # Compute R2
  y_true <- df_test$charges
  sse_sample <- sum((y_true - test_pred_sample)^2)
  sst <- sum((y_true - mean(y_true))^2)
  R2_samples[i] <- 1 - sse_sample/sst
}

# Print all 30 R2 values, one per line, with index
for (i in 1:30) {
  cat(sprintf("Tree %2d: R-squared on test set = %.4f\n", i, R2_samples[i]))
}
```

```
## Tree  1: R-squared on test set = -0.1308
## Tree  2: R-squared on test set = -0.0180
## Tree  3: R-squared on test set = -0.2456
## Tree  4: R-squared on test set = 0.0144
## Tree  5: R-squared on test set = -0.0797
## Tree  6: R-squared on test set = -0.1790
## Tree  7: R-squared on test set = 0.0681
## Tree  8: R-squared on test set = 0.0597
## Tree  9: R-squared on test set = 0.0632
## Tree 10: R-squared on test set = -0.3430
## Tree 11: R-squared on test set = -0.0013
## Tree 12: R-squared on test set = -0.0490
## Tree 13: R-squared on test set = -0.1315
## Tree 14: R-squared on test set = -0.3888
## Tree 15: R-squared on test set = -0.2100
## Tree 16: R-squared on test set = 0.0092
## Tree 17: R-squared on test set = 0.0037
## Tree 18: R-squared on test set = -0.0150
## Tree 19: R-squared on test set = -0.0479
## Tree 20: R-squared on test set = -0.0585
## Tree 21: R-squared on test set = -0.0672
## Tree 22: R-squared on test set = -0.0909
## Tree 23: R-squared on test set = -0.0463
## Tree 24: R-squared on test set = -0.0486
## Tree 25: R-squared on test set = -0.2147
## Tree 26: R-squared on test set = -0.1230
## Tree 27: R-squared on test set = 0.0071
## Tree 28: R-squared on test set = 0.0701
## Tree 29: R-squared on test set = -0.0299
## Tree 30: R-squared on test set = -0.0682
```

```r
# Print basetree R2 for comparison
cat(sprintf("\nBasetree: R-squared on test set = %.4f\n", R2_basetree))
```

```
##
## Basetree: R-squared on test set = 0.0557
```

```r
# Plot histogram of the 30 R2 values
hist(R2_samples, breaks = 10, main = "Histogram of R2 on Test Set (30 Sample Trees)",
     xlab = "R-squared", col = "orange", border = "white")
abline(v = R2_basetree, col = "blue", lwd = 2, lty = 2)
legend("topright", legend = "Base Tree R2", col = "blue", lwd = 2, lty = 2)
grid()
```



Results

Across the 30 bootstrapped trees (each trained on a random sample of 50 points with replacement, max depth = 4), the test-set R^2 values ranged from about -0.39z to +0.07, with most values negative. This means the bootstrapped trees generally performed worse than predicting the mean of the response. In contrast, the basetree (trained on the full training data with the same depth limit) achieved a small but positive R^2 of 0.0557, indicating that access to the full dataset led to slightly better generalization. A histogram of the 30 values would show them tightly clustered around negative performance, while the basetree stands just above zero.

The poor results arise mainly from two factors.

1. First, using only 50 observations in each bootstrap sample provides too little information, producing highly variable and weak models. Because sampling is done with replacement, some observations appear multiple times while others are skipped, reducing even further the variety of data the model sees.
2. Second, limiting tree depth to 4 constrains model complexity, so the trees cannot capture important structure in the data.

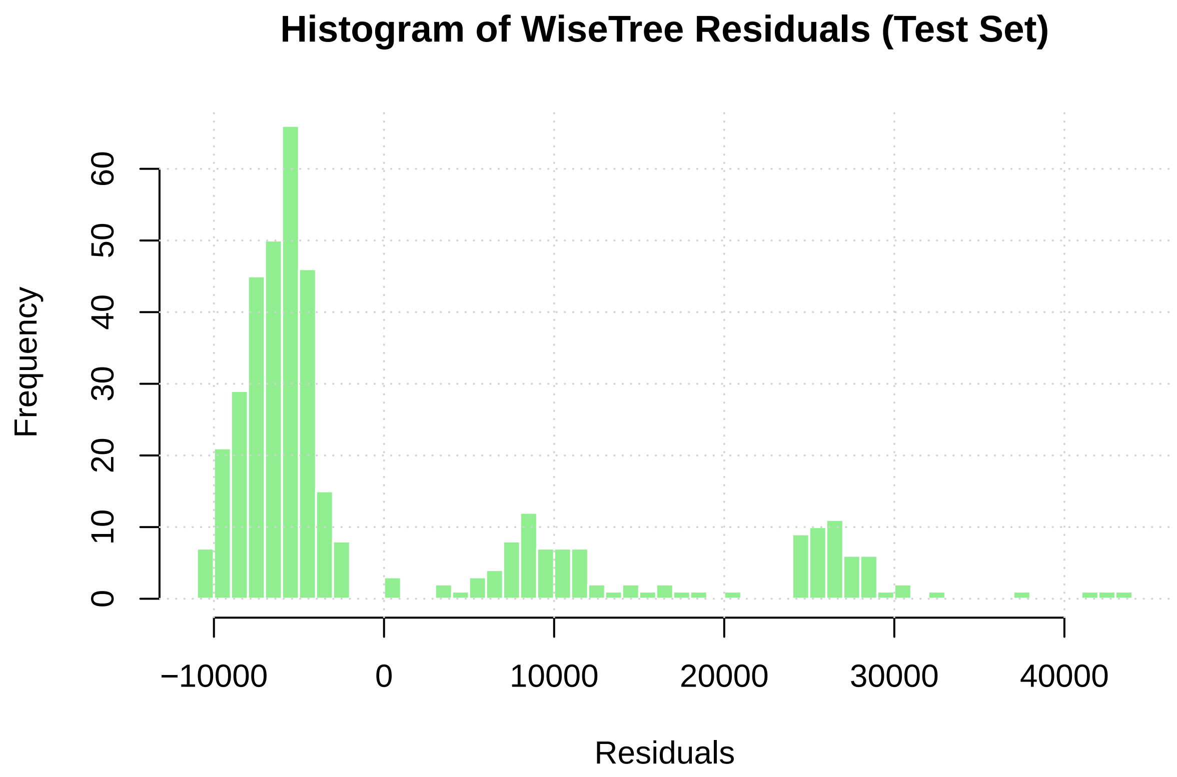
Question (f)

Now we are going to create `WiseTree`, an ensemble of 30 trees, each trained on a different bootstrap sample of size 50 (with replacement) from the training set, and each with max depth 4. In this ensemble, predictions are made by averaging the predictions of all 30 trees.

```r
# Use the 30 pretrained trees from wise_tree_models

# Make predictions by averaging the predictions of all trees (WiseTree ensemble)
wise_tree_pred <- rowMeans(sapply(wise_tree_models, function(model)
                                  predict(model, newdata = df_test)))

# Calculate residuals for WiseTree
wise_tree_residuals <- df_test$charges - wise_tree_pred

# Summarize the residuals
wise_tree_residuals_summary <- summary(wise_tree_residuals)
print(wise_tree_residuals_summary)
```

```
##     Min.   1st Qu.   Median     Mean  3rd Qu.     Max.
## -10977.0  -7022.2  -5260.9    435.2   7110.7  43943.3
```

```r
# Plot histogram of the residuals
hist(wise_tree_residuals, breaks = 40, main = "Histogram of WiseTree Residuals (Test Set)",
     xlab = "Residuals", col = "lightgreen", border = "white")
grid()
```



Question (g)

```r
# Calculate R-squared for WiseTree on the test set
y_true <- df_test$charges
sse_wisetree <- sum((y_true - wise_tree_pred)^2)
sst <- sum((y_true - mean(y_true))^2)
R2_wisetree <- 1 - sse_wisetree/sst
cat(sprintf("WiseTree: R-squared on test set = %.4f\n", R2_wisetree))
```

```
## WiseTree: R-squared on test set = 0.0950
```

```r
# Compare with basetree R2
cat(sprintf("Basetree: R-squared on test set = %.4f\n", R2_basetree))
```

```
## Basetree: R-squared on test set = 0.0557
```

WiseTree outperforms the basetree because it averages predictions from 30 bootstrapped trees, giving it more stability and better generalization. Bootstrapping samples the training set with replacement, so each tree sees a slightly different dataset, introducing diversity among the models. Averaging across these diverse but shallow trees reduces variance, making WiseTree more reliable than a single tree trained on the full dataset.