House Price Prediction – Random Forest Model

Tian

Summary

Preprocessing

Transformation (if any)

- price: performed box-cox transformation on the response variate price.
- saledate: transformed as the number of days since 1970/01/01, then take the power of 5
- landarea: take log
- stories: take log
- rmdl_diff: log(rmdl_diff + 1) because some of them are zero
- all categorical variables is treated as factors
- level of `cndtn` is specified
- matched the level of `dtest` and `dtrain`

New Variables

- if_rmdl : indicator whether the house is remodeled
- rmdl_diff: numerical rep the difference between remodel date and sale date; 0 if remodel is after sale
- saleyear: year of the sale
- buy_first: indicator whether the house is build first or buy first
- total_bath: combine number of bathroom and number of half-bathrooms (bathrm+0.5hf_bathrm)
- build_age: the years between sold year and year of build
- avg_room_size: divide `gba` by (the number of rooms + 1) to get the average size of rooms
- inter1: interaction term between latitude and saledate
- inter2: interaction term between longitude and saledate
- inter3: interaction term between gba and saledate
- inter4: interaction term between landarea and longitude
- inter5: interaction term between eyb and ayb
- inter6: interaction term between build_age and latitude

Model Building/Tuning

Main package used: `randomforest`, `ranger`, `caret`

Parameters tuned and their optimal values:

- mtry: 37
- min.node.size: 5
- splitrule: extratrees

final model: ranger($\frac{(price^{0.101}-1)}{0.101} \sim \sim .$, data = dtrain_full, mtry = 37, splitrule = "extratrees", min.node.size = 5)

1.Preprocessing

1.1 Loading data

```
load("RF.Rdata")
```

1.2 missing values

Check missing values for each predictor:

```
colSums(is.na(dtrain))
```

```
##     bathrm  hf_bathrm       heat         ac      rooms      bedrm        ayb
##          0          0          0          0          0          0         17
##    yr_rmdl        eyb    stories   saledate      price        gba      style
##       2410          0          4          0          0          0          0
##      grade      cndtn    extwall       roof    intwall   kitchens fireplaces
##          0          0          0          0          0          1          0
##   landarea   latitude  longitude       nbhd       ward   quadrant
##          0          0          0          0          0         32
```

```
colSums(is.na(dtest))
```

```
##         Id     bathrm  hf_bathrm       heat         ac      rooms      bedrm
##          0          0          0          0          0          0          0
##        ayb    yr_rmdl        eyb    stories   saledate        gba      style
##          3        390          0          0          0          0          0
##      grade      cndtn    extwall       roof    intwall   kitchens fireplaces
##          0          0          0          0          0          0          0
##   landarea   latitude  longitude       nbhd       ward   quadrant
##          0          0          0          0          0          4
```

So far we don't deal with missing values in `yr_rmdl`, we will add two new variables later to explain it so `yr_rmdl` won't be used directly in the model.

```
# =========== train ===============
avg_gap_train <- mean(dtrain$eyb-dtrain$ayb, na.rm = T)
# missing ayb is recoded as eyb - avg_gap between ayb and eyb
dtrain$ayb <- ifelse(is.na(dtrain$ayb), dtrain$eyb-avg_gap_train, dtrain$ayb)
# missing quadrant is recoded as "NW" bc "NW" is most popular
dtrain$quadrant <- ifelse(is.na(dtrain$quadrant), "NW", dtrain$quadrant)


#=========== test ============
avg_gap_test <- mean(dtest$eyb-dtest$ayb, na.rm = T)
# missing ayb is recoded as eyb - avg_gap between ayb and eyb
dtest$ayb <- ifelse(is.na(dtest$ayb), dtest$eyb-avg_gap_test, dtest$ayb)
# missing quadrant is recoded as "NW" bc "NW" is most popular
dtest$quadrant <- ifelse(is.na(dtest$quadrant), "NW", dtest$quadrant)
```

```r
colSums(is.na(dtrain))
```

```
##     bathrm  hf_bathrm       heat         ac      rooms      bedrm        ayb
##          0          0          0          0          0          0          0
##    yr_rmdl        eyb    stories   saledate      price        gba      style
##       2410          0          4          0          0          0          0
##      grade      cndtn    extwall       roof    intwall   kitchens fireplaces
##          0          0          0          0          0          1          0
##   landarea   latitude  longitude       nbhd       ward   quadrant
##          0          0          0          0          0          0
```

```r
colSums(is.na(dtest))
```

```
##         Id     bathrm  hf_bathrm       heat         ac      rooms      bedrm
##          0          0          0          0          0          0          0
##        ayb    yr_rmdl        eyb    stories   saledate        gba      style
##          0        390          0          0          0          0          0
##      grade      cndtn    extwall       roof    intwall   kitchens fireplaces
##          0          0          0          0          0          0          0
##   landarea   latitude  longitude       nbhd       ward   quadrant
##          0          0          0          0          0          0
```

1.3 new variable

```r
# binary variable check whether the house is remodeled
dtrain$if_rmdl <- ifelse(is.na(dtrain$yr_rmdl), 0, 1)
dtrain$if_rmdl <- as.factor(dtrain$if_rmdl)

dtest$if_rmdl <- ifelse(is.na(dtest$yr_rmdl), 0, 1)
dtest$if_rmdl <- as.factor(dtest$if_rmdl)


# year of the house sold
dtrain$saleyear<-as.numeric(substr(dtrain$saledate, 1, 4))
dtest$saleyear<-as.numeric(substr(dtest$saledate, 1, 4))


# the difference between sale year and the remodel year, if remodel is after sale
# then 0
for (i in seq(nrow(dtrain))) {
  if (is.na(dtrain$yr_rmdl[i])) {
    dtrain$rmdl_diff[i] <- 0
  } else if (dtrain$saleyear[i] <= dtrain$yr_rmdl[i]) {
    dtrain$rmdl_diff[i] <- 0
  } else if (dtrain$saleyear[i] > dtrain$yr_rmdl[i])
    dtrain$rmdl_diff[i] <- dtrain$saleyear[i] - dtrain$yr_rmdl[i]
}

for (i in seq(nrow(dtest))) {
    if (is.na(dtest$yr_rmdl[i])) {
      dtest$rmdl_diff[i] <- 0
```

```
    } else if (dtest$saleyear[i] <= dtest$yr_rmdl[i]) {
      dtest$rmdl_diff[i] <- 0
    } else if (dtest$saleyear[i] > dtest$yr_rmdl[i])
      dtest$rmdl_diff[i] <- dtest$saleyear[i] - dtest$yr_rmdl[i]
}

# average room size
dtrain$avg_room_size <- dtrain$gba / (dtrain$rooms +1)
dtest$avg_room_size <- dtest$gba / (dtest$rooms +1)

# year since build
dtrain$build_age <- as.numeric(substr(dtrain$saledate, 1,4)) - dtrain$ayb
dtest$build_age <- as.numeric(substr(dtest$saledate, 1,4)) - dtest$ayb


# combine bathroom and half_bathroom
dtrain$total_bath <- dtrain$bathrm+0.5*dtrain$hf_bathrm
dtest$total_bath <- dtest$bathrm+0.5*dtest$hf_bathrm

# whether it's sold first or build first
dtrain$buy_first <- as.factor(as.numeric(dtrain$saleyear < dtrain$ayb))
dtest$buy_first <- as.factor(as.numeric(dtest$saleyear < dtest$ayb))
```

```
# fill the na for yr_rmdl as 0
dtrain$yr_rmdl <- ifelse(is.na(dtrain$yr_rmdl), 0, dtrain$yr_rmdl)
dtest$yr_rmdl <- ifelse(is.na(dtest$yr_rmdl), 0, dtest$yr_rmdl)
# remove haft bathrooom
dtrain <- dtrain[, -which(names(dtrain) == "hf_bathrm")]
dtest <- dtest[, -which(names(dtest) == "hf_bathrm")]
# remove yr_rmdl
dtrain <- dtrain[, -which(names(dtrain) == "yr_rmdl")]
dtest <- dtest[, -which(names(dtest) == "yr_rmdl")]
```

```
dtrain_full <- na.omit(dtrain)
colSums(is.na(dtrain_full))
```

```
##        bathrm          heat            ac         rooms         bedrm
##             0             0             0             0             0
##           ayb           eyb       stories      saledate         price
##             0             0             0             0             0
##           gba         style         grade         cndtn       extwall
##             0             0             0             0             0
##          roof       intwall      kitchens    fireplaces      landarea
##             0             0             0             0             0
##      latitude     longitude          nbhd          ward      quadrant
##             0             0             0             0             0
##       if_rmdl      saleyear     rmdl_diff avg_room_size     build_age
##             0             0             0             0             0
##    total_bath     buy_first
##             0             0
```

```r
colSums(is.na(dtest))
```

```
##            Id        bathrm          heat            ac         rooms
##             0             0             0             0             0
##        bedrm           ayb           eyb       stories      saledate
##             0             0             0             0             0
##          gba         style         grade         cndtn       extwall
##             0             0             0             0             0
##         roof       intwall      kitchens    fireplaces      landarea
##             0             0             0             0             0
##     latitude     longitude          nbhd          ward      quadrant
##             0             0             0             0             0
##       if_rmdl      saleyear     rmdl_diff avg_room_size     build_age
##             0             0             0             0             0
##    total_bath     buy_first
##             0             0
```

Now, `dtrain_full` is the complete data frame we will be working with.

1.3 data transformation

```r
# the number of days since 1970/01/01
dtrain_full$saledate <- as.numeric(as.Date(dtrain_full$saledate))
dtest$saledate <- as.numeric(as.Date(dtest$saledate))


dtrain_full[] <- lapply(dtrain_full, function(x) if(is.character(x)) factor(x) else x)
dtest[] <- lapply(dtest, function(x) if(is.character(x)) factor(x) else x)
```

```r
condition_levels <- c("Poor", "Fair", "Average", "Good", "Very Good", "Excellent")
dtrain_full$cndtn <- factor(dtrain_full$cndtn, levels = condition_levels)
dtest$cndtn <- factor(dtest$cndtn, levels = condition_levels)
```

```r
dtrain_full$gba <- log(dtrain_full$gba)
dtrain_full$landarea <- log(dtrain_full$landarea)
dtrain_full$saledate <- dtrain_full$saledate^5
dtrain_full$stories <- log(dtrain_full$stories)
dtrain_full$rmdl_diff <- log(dtrain_full$rmdl_diff + 1)
dtrain_full$price <- (dtrain_full$price^0.101-1)/0.101
```

```r
# match the level of dtrain and dtest
for (var in names(dtrain_full)) {
    if (is.factor(dtrain_full[[var]]) && is.factor(dtest[[var]])) {
      updated_levels <- setdiff(levels(dtest[[var]]), levels(dtrain_full[[var]]))
      if (length(updated_levels) > 0) {
        common <- names(sort(table(dtrain_full[[var]]), decreasing = TRUE))[1]
        levels(dtest[[var]]) <- union(levels(dtrain_full[[var]]), levels(dtest[[var]]))
        dtest[[var]][dtest[[var]] %in% updated_levels] <- common
        dtest[[var]] <- factor(dtest[[var]], levels = levels(dtrain_full[[var]]))
      }
```

```
    }
  }
```

```
# add interaction terms
dtrain_full$inter1 <- with(dtrain_full, latitude * saledate)
dtrain_full$inter2 <- with(dtrain_full, longitude * saledate)
dtrain_full$inter3 <- with(dtrain_full, gba * saledate)
dtrain_full$inter4 <- with(dtrain_full, landarea * longitude)
dtrain_full$inter5 <- with(dtrain_full, eyb * ayb)
dtrain_full$inter6 <- with(dtrain_full, build_age * latitude)

dtest$inter1 <- with(dtest, latitude * saledate)
dtest$inter2 <- with(dtest, longitude * saledate)
dtest$inter3 <- with(dtest, gba * saledate)
dtest$inter4 <- with(dtest, landarea * longitude)
dtest$inter5 <- with(dtest, eyb * ayb)
dtest$inter6 <- with(dtest, build_age * latitude)
```

2. Model building

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ranger)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ranger':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Split test and training data

```
N <- nrow(dtrain_full)
N_train <- round(2* N /3)
N_test <- N - N_train
id.train <- sample(1:N, N_train, replace=FALSE)
id.test <- setdiff(1:N, id.train)
test <- dtrain_full[id.test,]
```

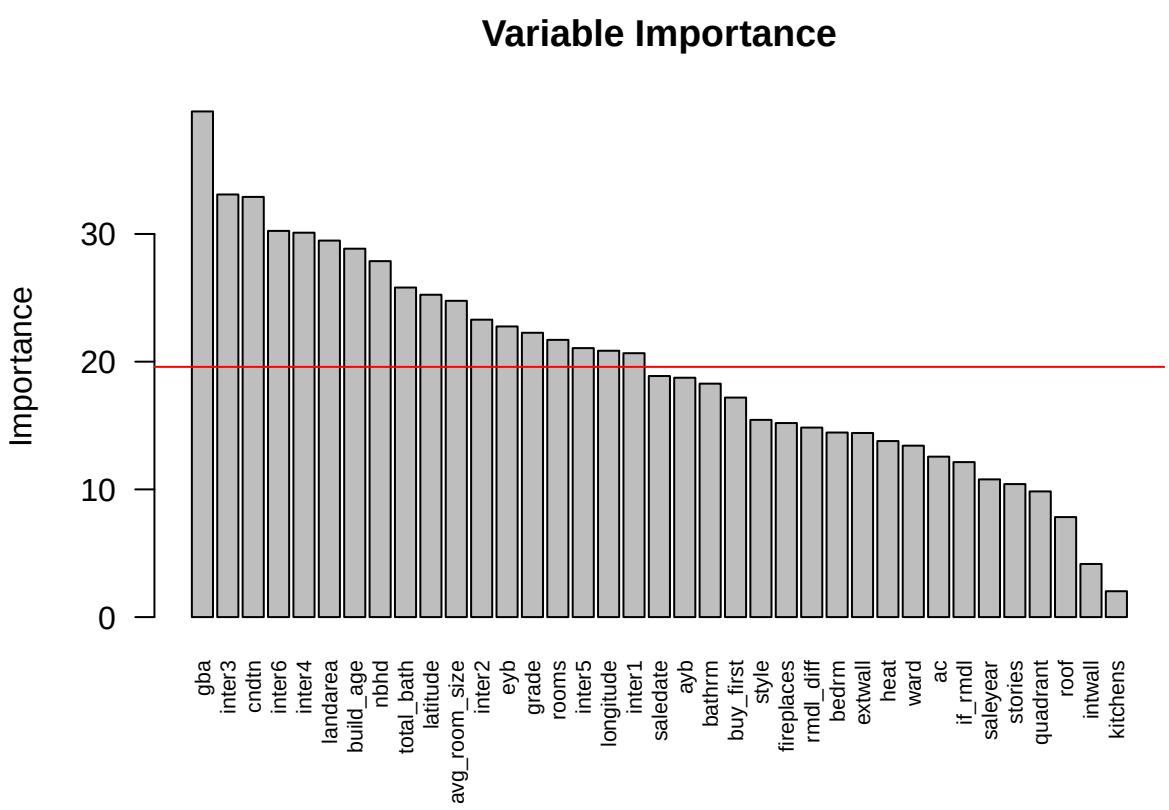
```
get.newdata <- function(fittedTree, test.data){
  f <- formula(fittedTree)
  as.list(test.data[,attr(terms(f), "term.labels")])
}
#
# And a similar function that will extract the response values
# This is kind of hairy, formula manipulation ... feel free to ignore ...
get.response <- function(fittedTree, test.data){
  f <- formula(fittedTree)
  terms <- terms(f)
  response.id <- attr(terms, "response")
  response <- as.list(attr(terms, "variables"))[[response.id + 1]]
  with(test.data,eval(response))
}

get.explanatory_varnames <- function(formula){
  f <- as.formula(formula)
  terms <- terms(f)
  attr(terms, "term.labels")
}
```

2.1 The naive model

```
set.seed(444)
naive <- randomForest(price ~ ., data=dtrain_full,
                      importance=TRUE)
```

We plot the importance measured by mean decrease in accuracy

```
importance <- importance(naive, type = 1)
importance_sort <- sort(importance, decreasing = TRUE)
indices <- order(-importance)
names <- row.names(importance)[indices]
barplot(importance_sort, names.arg = names, las = 2,
        main = "Variable Importance",
        ylab = "Importance", cex.names = 0.7)
avg <- mean(importance)
abline(h = avg, col = "red")
```



Remove the last two predictors

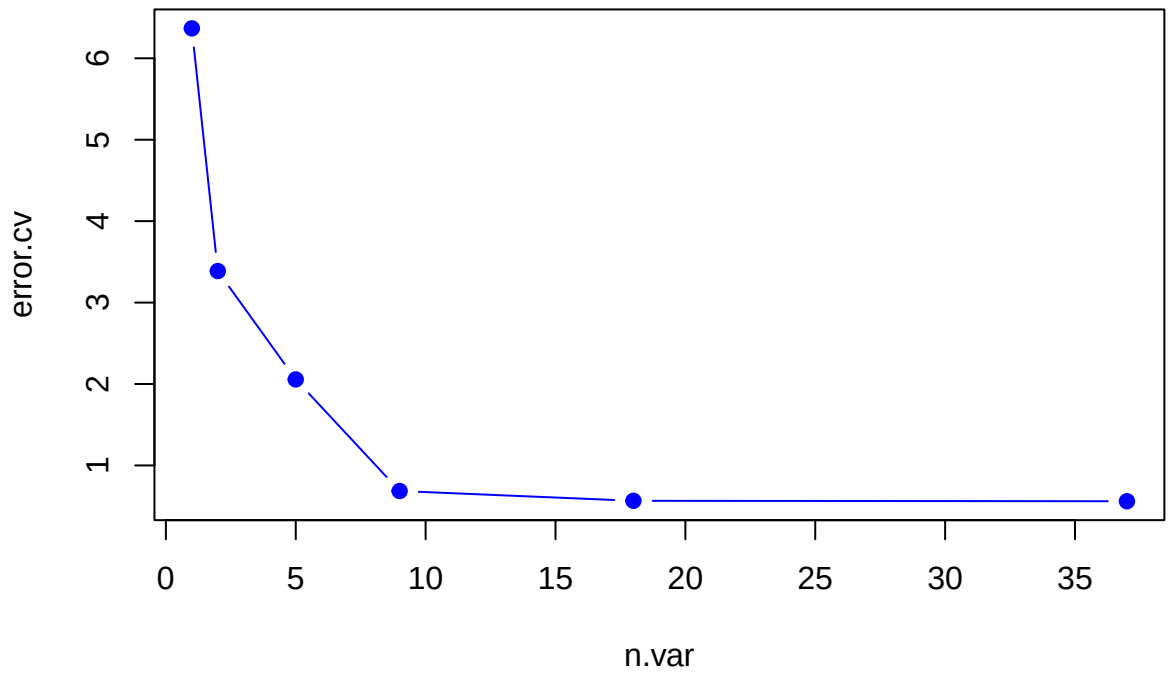
```
select_var <- names[1:35]
select_var
```

```
##  [1] "gba"           "inter3"        "cndtn"         "inter6"
##  [5] "inter4"        "landarea"      "build_age"     "nbhd"
##  [9] "total_bath"    "latitude"      "avg_room_size" "inter2"
## [13] "eyb"           "grade"         "rooms"         "inter5"
## [17] "longitude"     "inter1"        "saledate"      "ayb"
## [21] "bathrm"        "buy_first"     "style"         "fireplaces"
## [25] "rmdl_diff"     "bedrm"         "extwall"       "heat"
## [29] "ward"          "ac"            "if_rmdl"       "saleyear"
## [33] "stories"       "quadrant"      "roof"
```

Run cross-validation on the naive model

```
trainy <- dtrain_full[,"price"]
trainx <- dtrain_full[, get.explanatory_varnames(naive)]
cv_naive <- rfcv(trainx = trainx, trainy = trainy, cv.fold = 5)
```

```
with(cv_naive, plot(n.var, error.cv, pch = 19, type="b", col="blue"))
```



We see the error stops decreasing after the the number of predictors hits 17. Since more predictors don't increase error, we use all the predictors in the model.

2.2 tuning

```
set.seed(444)
train_control <- trainControl(method="cv", number=5)
tuneGrid <- expand.grid(.mtry=c(8, 9, 10, 14, 20, 30, 37),
                        .min.node.size = c(5, 6, 7, 9, 10),
                        .splitrule = c("variance", "extratrees"))

tr1 = train(
  x = dtrain_full[ , names(dtrain_full) != 'price'],
  y = dtrain_full[ , names(dtrain_full) == 'price'],
  method = 'ranger', trControl = train_control, tuneGrid = tuneGrid
)

tr1
```

```
## Random Forest
##
## 5995 samples
##   37 predictor
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 4795, 4796, 4796, 4796, 4797
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  splitrule   RMSE       Rsquared   MAE
##    8     5             variance    0.7650306  0.9385883  0.5160394
##    8     5             extratrees  0.7916866  0.9364473  0.5341729
##    8     6             variance    0.7650834  0.9386675  0.5156475
##    8     6             extratrees  0.7939805  0.9361423  0.5366892
##    8     7             variance    0.7646388  0.9387247  0.5160998
##    8     7             extratrees  0.7931330  0.9364105  0.5362267
##    8     9             variance    0.7672889  0.9382896  0.5182349
##    8     9             extratrees  0.7981975  0.9357201  0.5400046
##    8    10             variance    0.7672681  0.9383350  0.5176271
##    8    10             extratrees  0.8029750  0.9349679  0.5445453
##    9     5             variance    0.7618644  0.9389705  0.5149881
##    9     5             extratrees  0.7824729  0.9375733  0.5285091
##    9     6             variance    0.7637229  0.9387165  0.5156912
##    9     6             extratrees  0.7854039  0.9371402  0.5307732
##    9     7             variance    0.7624986  0.9389224  0.5155175
##    9     7             extratrees  0.7866535  0.9370422  0.5319972
##    9     9             variance    0.7654186  0.9384801  0.5170265
##    9     9             extratrees  0.7936229  0.9360759  0.5360348
##    9    10             variance    0.7667731  0.9382840  0.5175638
##    9    10             extratrees  0.7955790  0.9357815  0.5379907
##   10     5             variance    0.7624568  0.9388105  0.5145106
##   10     5             extratrees  0.7808069  0.9375084  0.5261367
##   10     6             variance    0.7623725  0.9388377  0.5147804
##   10     6             extratrees  0.7813601  0.9374699  0.5279695
##   10     7             variance    0.7635438  0.9386552  0.5161056
##   10     7             extratrees  0.7834527  0.9372076  0.5303616
##   10     9             variance    0.7644969  0.9385125  0.5169397
##   10     9             extratrees  0.7886734  0.9365372  0.5330921
##   10    10             variance    0.7661767  0.9382747  0.5186602
##   10    10             extratrees  0.7892312  0.9365113  0.5328710
##   14     5             variance    0.7628746  0.9384741  0.5159876
##   14     5             extratrees  0.7669654  0.9389983  0.5186468
##   14     6             variance    0.7633494  0.9384002  0.5175770
##   14     6             extratrees  0.7659347  0.9392211  0.5173019
##   14     7             variance    0.7635652  0.9383994  0.5173595
##   14     7             extratrees  0.7673179  0.9390689  0.5191949
##   14     9             variance    0.7656623  0.9380378  0.5178532
##   14     9             extratrees  0.7698121  0.9388269  0.5214932
##   14    10             variance    0.7664240  0.9379329  0.5192694
##   14    10             extratrees  0.7732276  0.9382823  0.5233853
##   20     5             variance    0.7675110  0.9375245  0.5200794
##   20     5             extratrees  0.7540348  0.9406276  0.5102885
##   20     6             variance    0.7667267  0.9376700  0.5195267
##   20     6             extratrees  0.7539423  0.9407078  0.5113789
##   20     7             variance    0.7682138  0.9374046  0.5214199
##   20     7             extratrees  0.7552668  0.9405626  0.5117235
##   20     9             variance    0.7693957  0.9372284  0.5217651
##   20     9             extratrees  0.7597474  0.9398612  0.5151007
##   20    10             variance    0.7690786  0.9372998  0.5220148
```

```
##   20    10            extratrees  0.7599177  0.9399006  0.5155050
##   30     5            variance    0.7763646  0.9358603  0.5273921
##   30     5            extratrees  0.7428257  0.9420771  0.5054960
##   30     6            variance    0.7787880  0.9354657  0.5289665
##   30     6            extratrees  0.7438567  0.9419672  0.5059127
##   30     7            variance    0.7779420  0.9355953  0.5289066
##   30     7            extratrees  0.7443017  0.9419465  0.5061593
##   30     9            variance    0.7774582  0.9356640  0.5282376
##   30     9            extratrees  0.7480951  0.9413730  0.5087624
##   30    10            variance    0.7790027  0.9354263  0.5295631
##   30    10            extratrees  0.7475276  0.9414646  0.5084277
##   37     5            variance    0.7855580  0.9342224  0.5348830
##   37     5            extratrees  0.7404003  0.9424139  0.5031497
##   37     6            variance    0.7855503  0.9342305  0.5347623
##   37     6            extratrees  0.7416747  0.9421597  0.5039095
##   37     7            variance    0.7859677  0.9341671  0.5355118
##   37     7            extratrees  0.7409580  0.9423171  0.5044646
##   37     9            variance    0.7884747  0.9337452  0.5358030
##   37     9            extratrees  0.7432308  0.9419777  0.5056300
##   37    10            variance    0.7889609  0.9336481  0.5369761
##   37    10            extratrees  0.7447026  0.9418101  0.5074158
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 37, splitrule = extratrees
##  and min.node.size = 5.
```