# Course 2 Section 4.9 - YOUR TURN

Jiaying Wu

17/10/2020

```
# load library
library(tidyverse)
library(GGally)
library(rpart)

# load data
houses_raw <- read_csv("https://raw.githubusercontent.com/datascienceprogram/ids_course_data/master/Mell
```

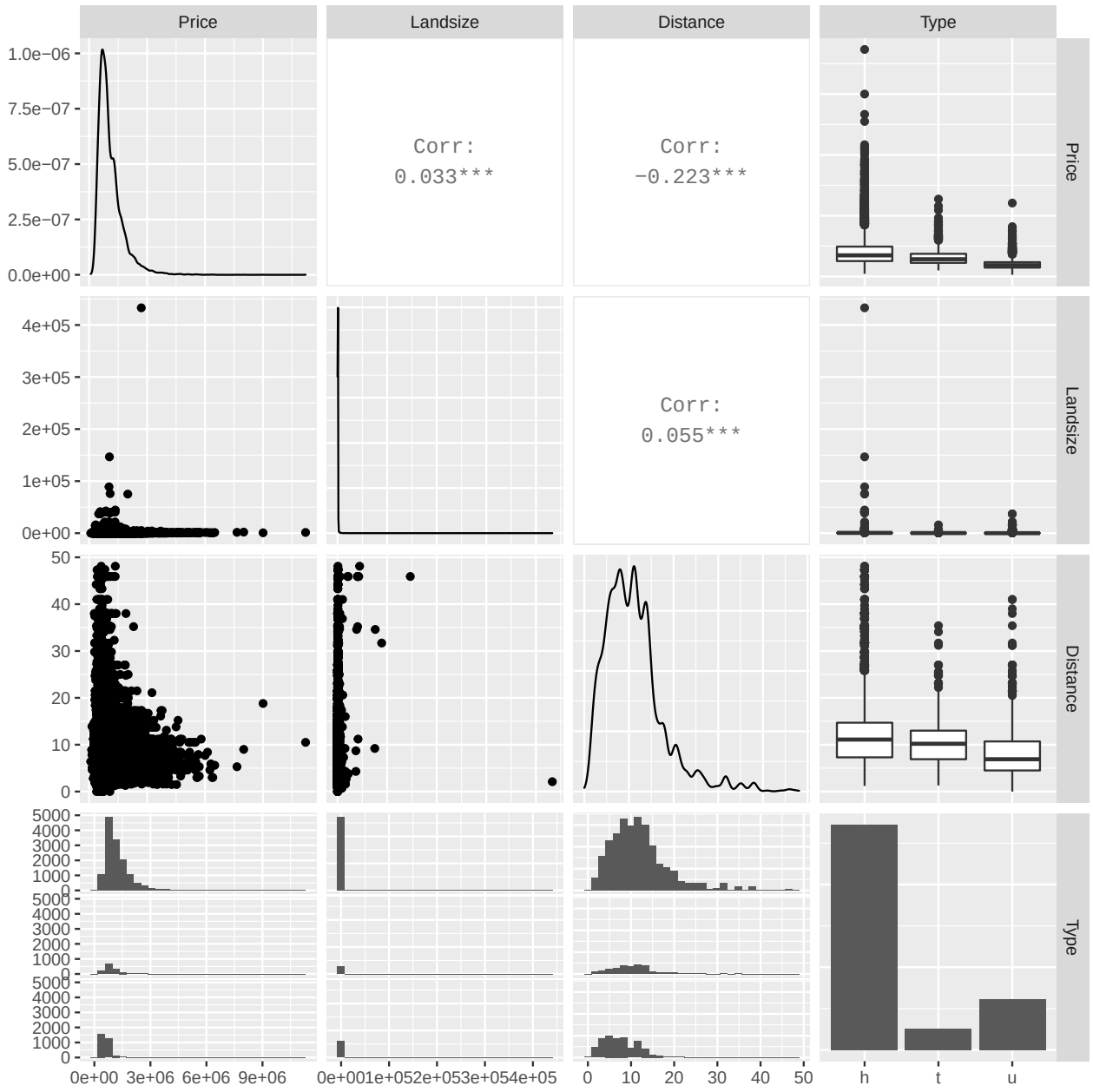
### Conduct a preliminary analysis

This data set contains 34857 observations and 21 variables. Among these variables, Landsize, Type, and Distance are of great interest because of their impact on price. A brief explanation of the variables:

- Landsize: the size of the land.
- Type: a house (h), townhouse (t), and unit (u).
- Distance: how far the property is from the Melbourne Central Business District (CBD).

### Produce a generalised pair plot

```
houses_raw %>%
  select(Price, Landsize, Distance, Type) %>%
  drop_na() %>%
  ggpairs()
```




now focuses on 11 variables instead of 21.

```
houses_full <- houses_raw %>%
  mutate(Suburb = as_factor(Suburb)) %>%
  filter(!is.na(Price)) %>%
  select(
    Price, Suburb, Rooms, Type, Distance, Bedroom2, Bathroom,
    Car, Landsize, YearBuilt, Propertycount
  )
```

**Is there any outlier in the data?**

```
houses_full %>%
  top_n(wt = Landsize, n = 10)
```

```
## # A tibble: 10 x 11
##     Price Suburb Rooms Type  Distance Bedroom2 Bathroom   Car Landsize YearBuilt
##     <dbl> <fct>  <dbl> <chr>    <dbl>    <dbl>    <dbl> <dbl>    <dbl>     <dbl>
##  1 2.00e6 Balwy~     3 h          9.2        3        1     2    75100        NA
##  2 5.72e5 Reser~     3 h         11.2        3        1     2    41400        NA
##  3 1.08e6 Silvan     3 h         34.6        3        2     2    76000        NA
##  4 2.70e6 Fitzr~     3 h          2.1        3        3     1   433014        NA
##  5 1.36e6 New G~     5 h         48.1        5        3     5    44500        NA
##  6 1.03e6 Wildw~     5 h         31.7        5        2     2    89030        NA
##  7 1.05e6 Bulle~     4 h         45.9        4        2     1   146699        NA
##  8 1.35e6 Gisbo~     4 h         45.9        4        2     8    40469      1990
##  9 1.15e6 Wandi~     4 h         35.2        4        2     2    40500        NA
## 10 8.65e5 Bulle~     4 h         45.9        4        2     0    42800      2000
## # ... with 1 more variable: Propertycount <dbl>
```

The house with land size in 433014 mostly likely to be an outlier

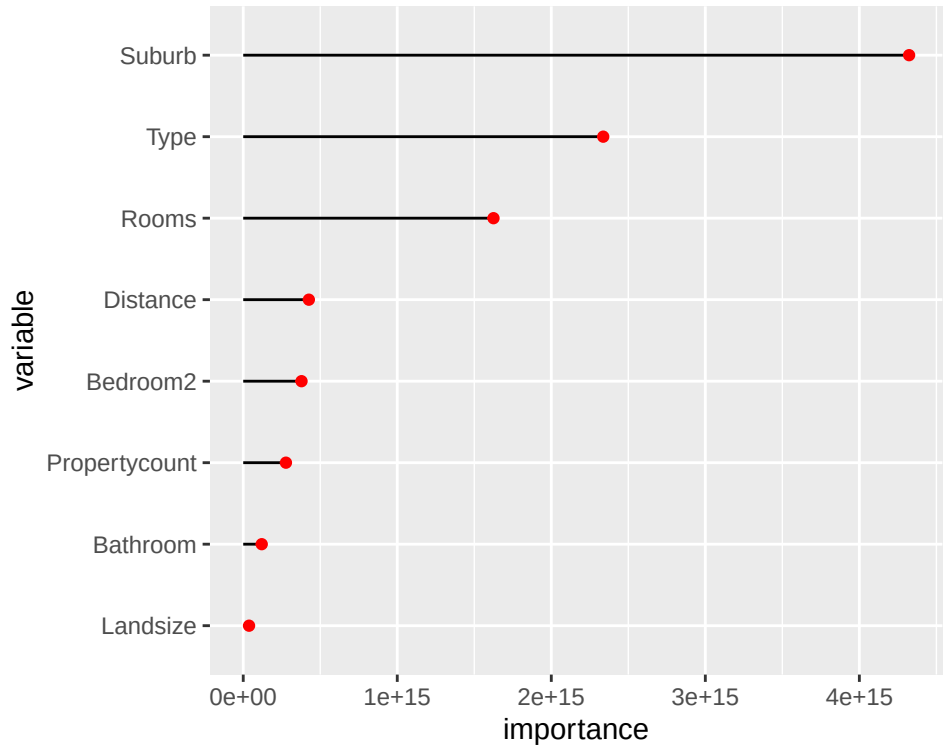
**Model fits**

```
houses_rp_fit <- rpart(Price ~ ., data = houses_full)
```

extract variable.importance, which contains information about variable importance from houses_rp_fit (the fitted model), and then plot it.

```
var_importance <- tibble(
  variable = names(houses_rp_fit$variable.importance),
  importance = houses_rp_fit$variable.importance)

var_importance %>%
  mutate(variable = fct_reorder(variable, importance)) %>%
  ggplot(aes(x = importance, y = variable)) +
  geom_segment(aes(x = 0, y = variable, xend = importance, yend = variable)) +
  geom_point(colour = "red")
```




remove the outliers and examine how the model results respond to the change.

```
houses_out <- houses_full %>%
  filter(Landsize != 433014)
```

### Working towards model accuracy

split the data to training (houses_train) and test sets (houses_test) in order to evaluate the model's accuracy.

```
set.seed(1234)
nobs <- nrow(houses_out)
idx <- sample(1:nobs, size = round(0.8 * nobs))
houses_train <- slice(houses_out, idx)
houses_test <- slice(houses_out, -idx)
```

train the tree model with the training set.

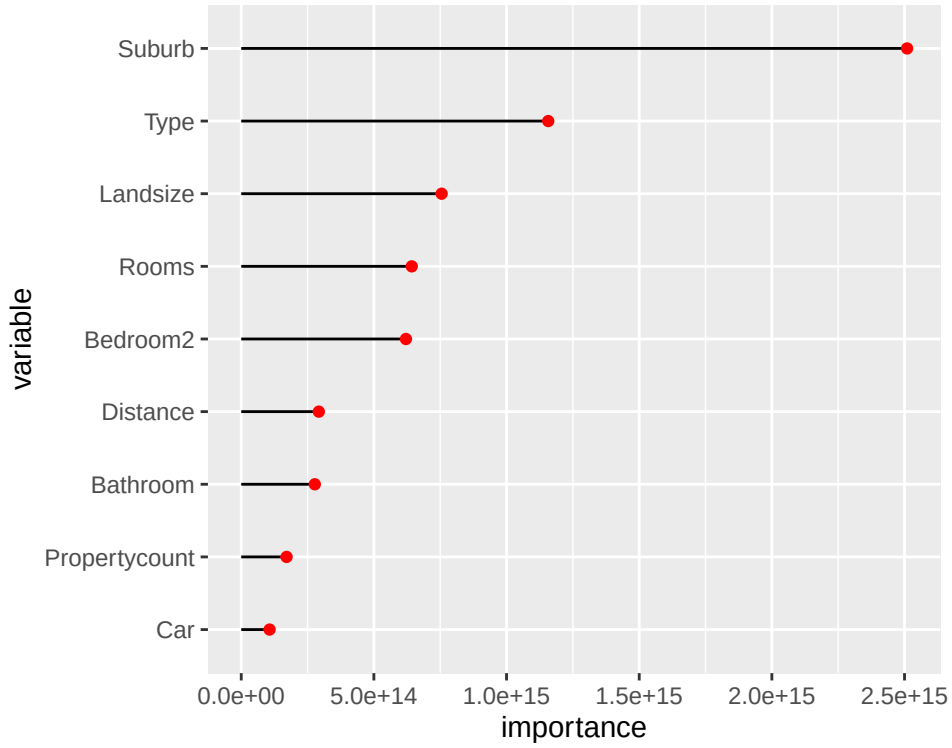
```
houses_rp_all <- rpart(Price ~ ., data = houses_train)
```

By removing the outlier(s), land size has moved from last place to third place in the ranking of important variables. This result is now consistent with our expectations.

```
var_importance <- tibble(
  variable = names(houses_rp_all$variable.importance),
  importance = houses_rp_all$variable.importance)

var_importance %>%
```

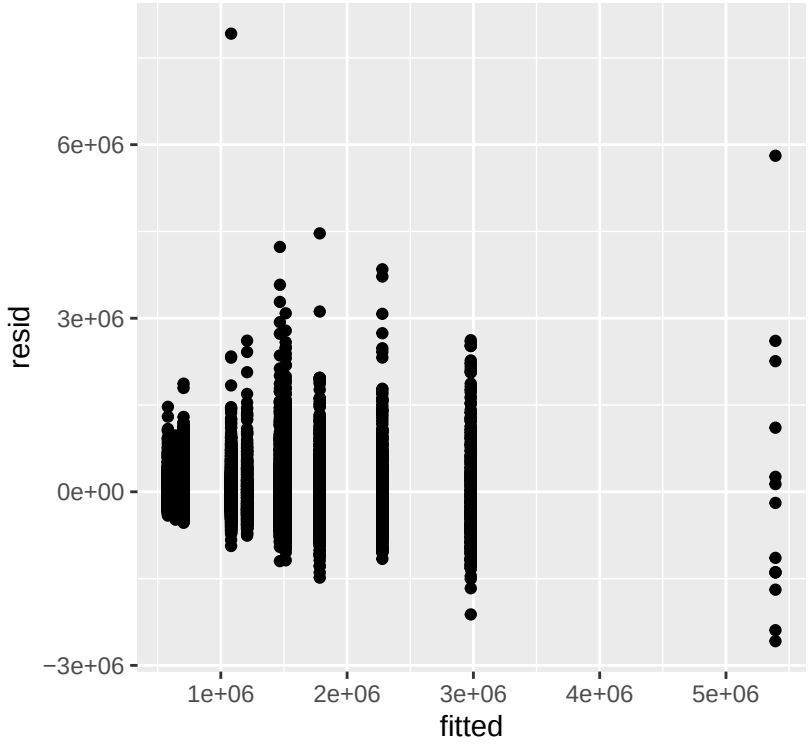
```
  mutate(variable = fct_reorder(variable, importance)) %>%
  ggplot(aes(x = importance, y = variable)) +
  geom_segment(aes(x = 0, y = variable, xend = importance, yend = variable)) +
  geom_point(colour = "red")
```




### Model diagnostics

Diagnostics tools are used to examine the residuals compared to fitted values.

```
tibble(
  resid = residuals(houses_rp_all),
  fitted = predict(houses_rp_all, newdata = houses_train)
) %>%
  ggplot(aes(x = fitted, y = resid)) +
  geom_point() +
  theme(aspect.ratio = 1)
```




### Compute the root mean squared error

compute the root mean squared error for both in-sample and out-of-sample predictions.

```
rmse <- function(actual, predicted) {
  sqrt(mean((predicted - actual)^2))
}
c("train" = rmse(houses_train$Price, predict(houses_rp_all, newdata = houses_train)),
  "test" = rmse(houses_test$Price, predict(houses_rp_all, newdata = houses_test)))
```

```
##    train     test
## 393844.8 395576.5
```

If the root mean squared error of test set lareger than train set, it might suggest the model is overfitted.

### Tune your parameters

To improve your prediction, you'll need to carry out some parameter tuning for the regression tree.

```
houses_rp_control <- rpart(Price ~ ., data = houses_train,
                           control = rpart.control(cp = 0.001))
c("train" = rmse(houses_train$Price, predict(houses_rp_control, newdata = houses_train)),
  "test" = rmse(houses_test$Price, predict(houses_rp_control, newdata = houses_test)))
```

```
##    train     test
## 323422.6 347785.3
```

**Q1.What aspect does the parameter cp control for the fit?**

cp is a complexity parameter, with anova splitting, this means that the overall R-squared must increase by cp at each step.

**Q2.Does cp = 0.001 complicate or simplify the default model?**

Complicate the default model.

**Q3.Does it give a better model fit?**

Yes, it have a lower rmse in both training set and test set.

**Q4.If we keep decreasing the cp value, what problem will we encounter?**

The model might overfitted, we might have a higher rmse in test set.