Lab Week 2

STAT5003

Contents

1 Melbourne house prices regression model **1**
- 1.1 Load the data 1
- 1.2 Initial data analysis 2
- 1.3 Finding association I 2
- 1.4 Finding association II 9
- 1.5 Impact of outliers 20
- 1.6 Prediction 21

Preparation and assumed knowledge

- Linear regression models covered in Module 2 content.

Aims

- Implementation of a linear regression using the `lm` function.
- Interpret the output of a linear model.
- Improve an existing linear model.
- Communicate your results.

1 Melbourne house prices regression model

In this section we will examine the dataset describing Melbourne house prices. This dataset was downloaded from Kaggle and the data was released under the CC BY-NC-SA 4.0 license. For this lab, we will focus on three subrubs - Brunswick, Craigieburn and Hawthorn and examine what variables or factors are associated with the housing price.

1.1 Load the data

Load the Melbourne house price dataset from Canvas.

Solution

```
melb.dat <- read.csv("Melbourne_housing_FULL.csv")
melbdata <- read_csv("Melbourne_housing_FULL.csv")
```

```
## Rows: 34857 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (8): Suburb, Address, Type, Method, SellerG, Date, CouncilArea, Regionname
## dbl (13): Rooms, Price, Distance, Postcode, Bedroom2, Bathroom, Car, Landsiz...
##
## 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.
```

1.2 Initial data analysis

We will need to subset the data to only look at 3 suburbs - Brunswick, Craigieburn and Hawthorn. Similar to lab 1, start the data analysis by generating some quantitative and graphical summaries. For example, determine the average price in each of these three suburbs. Explore more summaries of the data.

Solution

```
# Base R
melb.data.sub <- subset(melbdata, Suburb == "Hawthorn" | Suburb == "Brunswick" | Suburb == "Craigieburn
melb.data.sub2 <- subset(melbdata, Suburb %in% c("Hawthorn", "Brunswick", "Craigieburn"))
identical(melb.data.sub, melb.data.sub2)
```

```
## [1] TRUE
```

```
split.data <- split(melb.data.sub[["Price"]], melb.data.sub[["Suburb"]])
suburb.means <- vapply(split.data, mean, numeric(1L), na.rm = TRUE)
suburb.medians <- vapply(split.data, median, numeric(1L), na.rm = TRUE)
```

```
# Tidyverse way
melbdata.sub <- melbdata %>%
    filter(Suburb %in% c("Hawthorn", "Brunswick", "Craigieburn")) %>%
    mutate(Suburb = factor(Suburb, levels = c("Craigieburn", "Brunswick", "Hawthorn")))

melbdata %>%
  filter(Suburb %in% c("Hawthorn", "Brunswick", "Craigieburn")) %>%
  group_by(Suburb) %>%
  summarise(Mean_Price = mean(Price, na.rm = TRUE), Median_price = median(Price, na.rm = TRUE))
```

```
## # A tibble: 3 x 3
##   Suburb      Mean_Price Median_price
##   <chr>            <dbl>        <dbl>
## 1 Brunswick      977989.       950000
## 2 Craigieburn    566173.       562500
## 3 Hawthorn      1238074.       750500
```

For the following questions, use the subsetted data for the Suburbs of Brunswick, Craigieburn and Hawthorn.

1.3 Finding association I

To examine the association between house prices and a single variable, start by constructing a simple linear regression using only `BuildingArea` as a predictor. Use an appropriate statistic to justify the goodness of fit of the prediction and create a graphical output to enable you to assess your model fit.

Note: you might consider other variables too.

Solution Consider a scatter plot of `Price` against `BuildingArea` and overlay the prediction from the linear regression model.

```
# FORM THE LINEAR REGRESSION MODEL
lm1 <- lm(data = melbdata.sub, Price/1000 ~ BuildingArea)

# Inspect coefficients
coef(lm1)
```

```
##  (Intercept) BuildingArea
##   518.192115     3.800746
```

```
lm1 |> coef()#get coefficients
```

```
##  (Intercept) BuildingArea
##   518.192115     3.800746
```

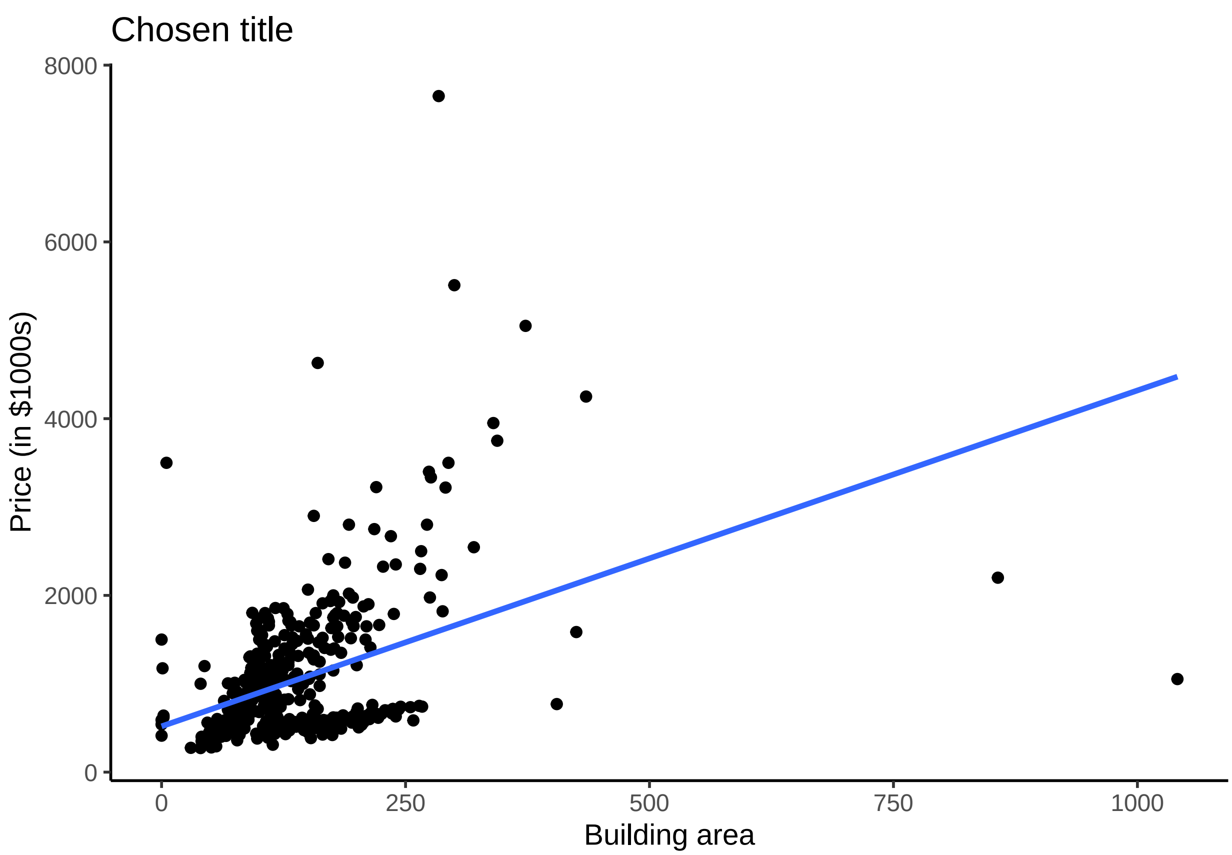
```
lm1 |> fitted() |> head() #fitted values
```

```
##         2         3         4         9        11        12
##  928.6727  871.6615 1312.5480 1609.0062  769.0413  670.2220
```

```
lm1 |> resid() |> head() #residuals ie errors, check mean is zero.
```

```
##          2          3          4          9         11         12
##   97.32732  930.83851  187.45198  620.99380 -359.04135 -397.72195
```

```
ggplot(melbdata.sub |> select(BuildingArea, Price) |> drop_na()) +
    aes(x = BuildingArea, y = Price/1000) +
    geom_point() + geom_smooth(formula = y ~ x, method = "lm", se = FALSE) +
    theme_classic() + labs(x = "Building area", y = "Price (in $1000s)", title = "Chosen title")
```

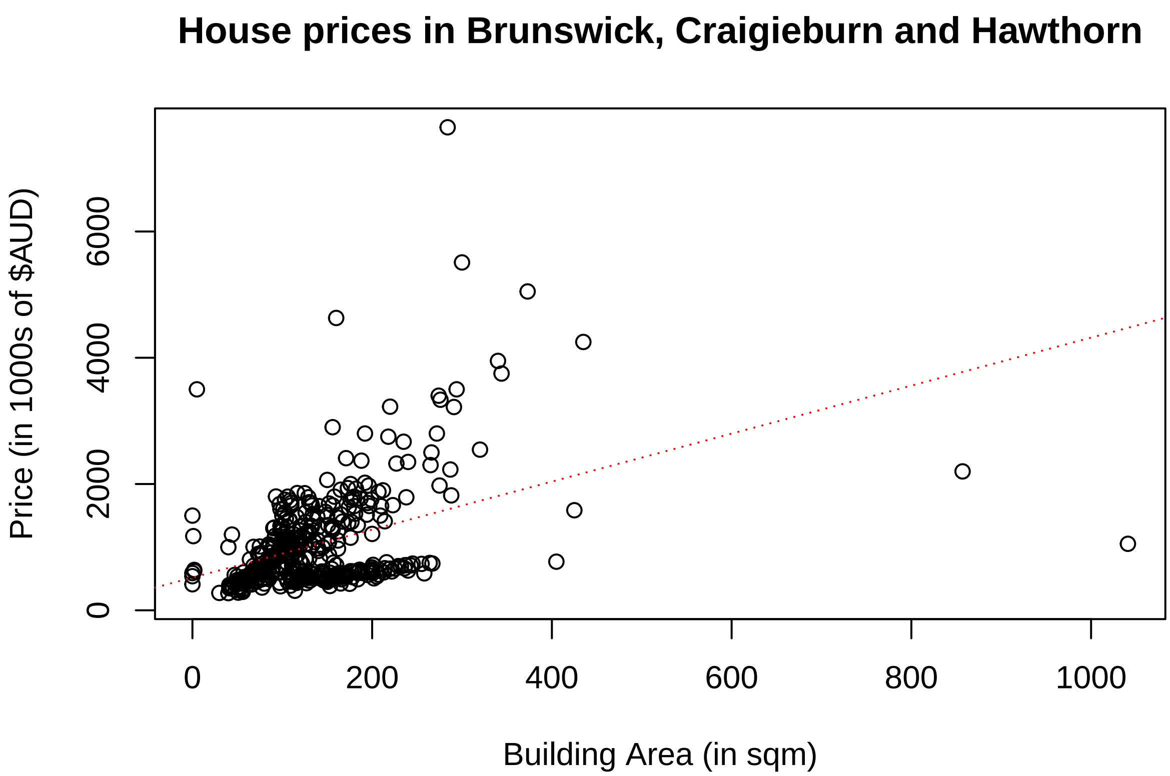



```
# Base R way
summary(lm1)
```

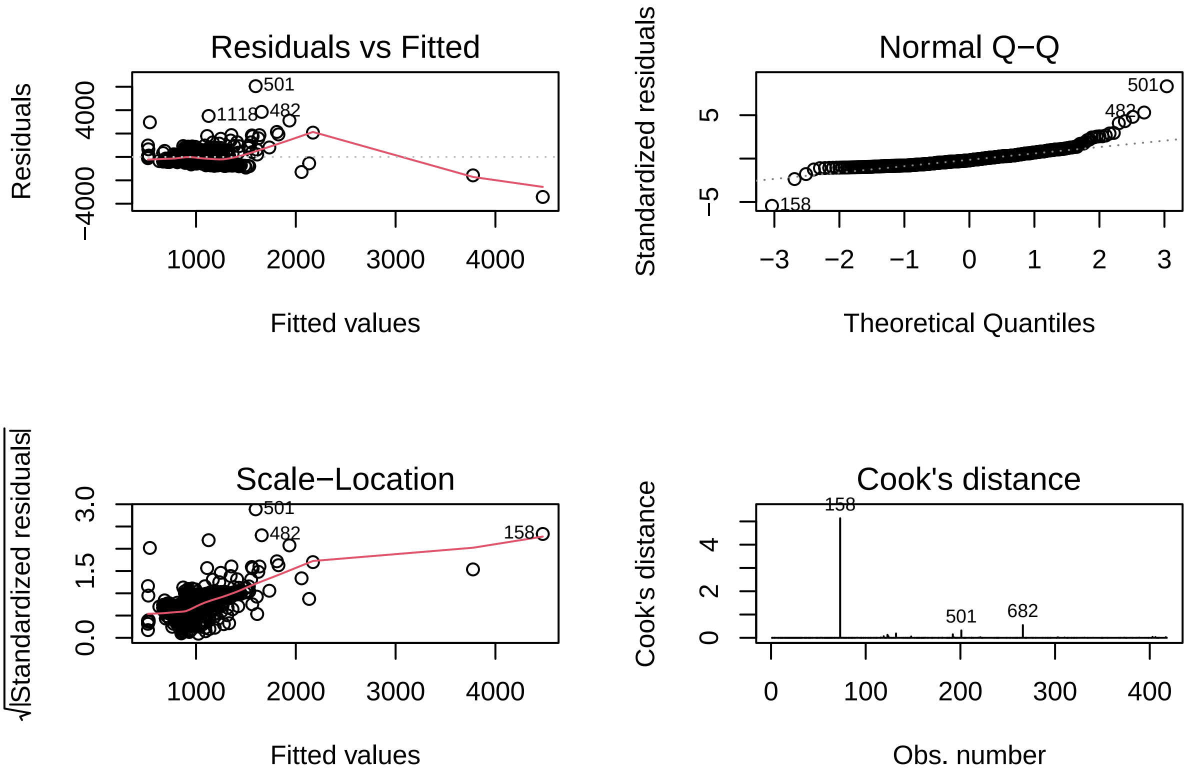
```
##
## Call:
## lm(formula = Price/1000 ~ BuildingArea, data = melbdata.sub)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3421.8  -463.9  -148.0   259.1  6052.4
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  518.1921    66.5956   7.781 5.74e-14 ***
## BuildingArea   3.8007     0.4082   9.311  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 730 on 416 degrees of freedom
##   (709 observations deleted due to missingness)
## Multiple R-squared:  0.1725, Adjusted R-squared:  0.1705
## F-statistic: 86.69 on 1 and 416 DF,  p-value: < 2.2e-16
```

```r
plot(Price/1000 ~ BuildingArea, data = melbdata.sub,
     main = "House prices in Brunswick, Craigieburn and Hawthorn",
     xlab = "Building Area (in sqm)", ylab = "Price (in 1000s of $AUD)")
abline(lm1, col = "red", lty = "dotted")
```




```r
par(mfrow = c(2, 2))
plot(lm1, which = 1:4)
```

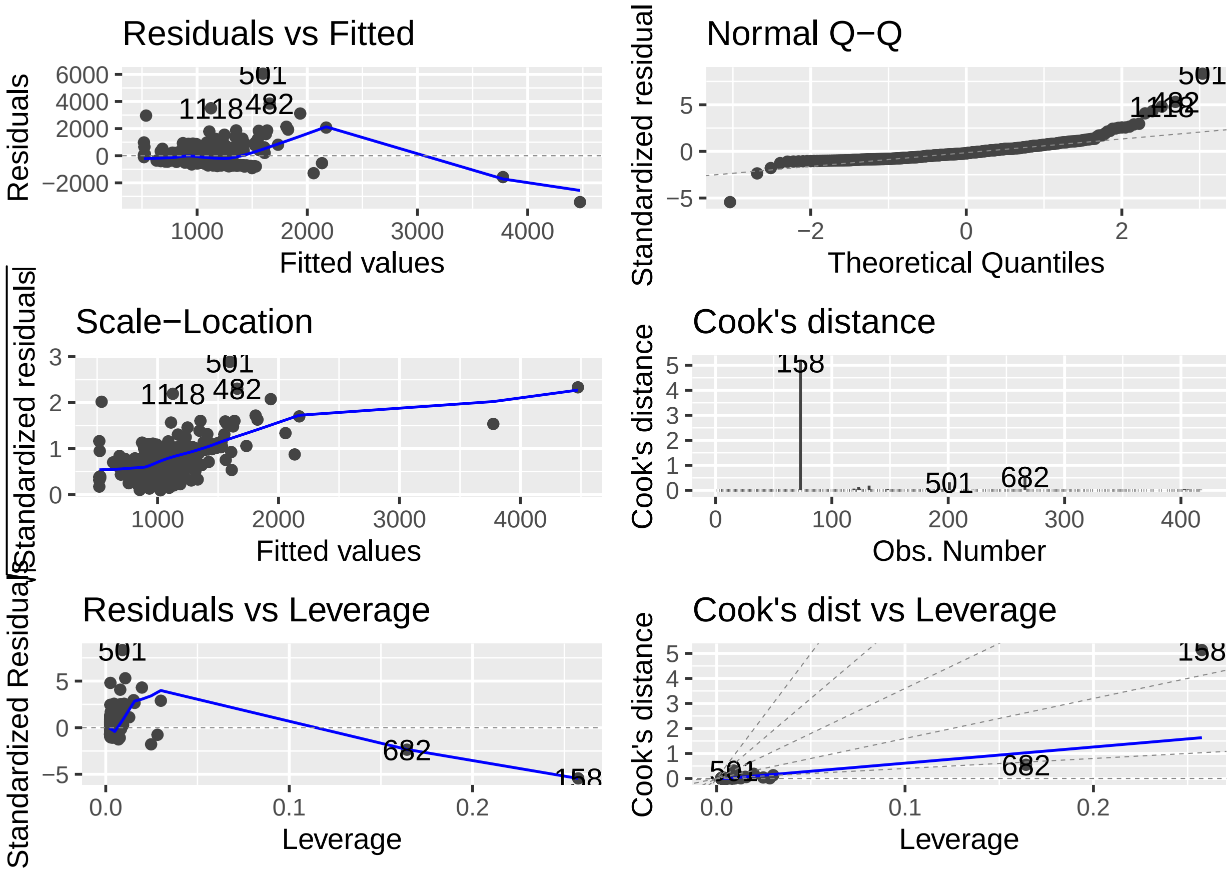



```r
r2 <- round(summary(lm1)$r.squared, 4)
r2
```

```
## [1] 0.1725
```

There is 17.25% of the variation in Price explained by the linear regression on Building Area.

```r
# Tidyverse way
library(ggfortify)
autoplot(lm1, which = 1:6, nrow = 3, ncol = 2)
```




```r
# LOTS OF NICE CONVENIENT CODE
# lm1 |> coefficients() #get coefficients
# lm1 |> fitted() #fitted values
# lm1 |> residuals() %>% mean()#residuals ie errors, check mean is zero.

library(broom)
lm1 |> augment() #full table of fitted values, cooks distance, etc
```

```
## # A tibble: 418 x 8
##    .rownames `Price/1000` BuildingArea .fitted    .hat .sigma    .cooksd .std.~1
##    <chr>            <dbl>        <dbl>   <dbl>   <dbl>  <dbl>      <dbl>   <dbl>
##  1 2                1026           108    929. 0.00267   731. 0.0000238   0.134
##  2 3                1802.           93    872. 0.00302   729. 0.00247     1.28
##  3 4                1500           209   1313. 0.00398   731. 0.000132    0.257
##  4 9                2230           287   1609. 0.00936   730. 0.00345     0.855
##  5 11                410            66    769. 0.00400   731. 0.000488   -0.493
##  6 12                272.           40    670. 0.00538   731. 0.000807   -0.546
##  7 13                680           100    898. 0.00284   731. 0.000128   -0.299
##  8 16                400            61    750. 0.00423   731. 0.000491   -0.481
##  9 17                950            96    883. 0.00294   731. 0.0000124   0.0918
## 10 20                860            97    887. 0.00291   731. 0.00000198 -0.0369
## # ... with 408 more rows, and abbreviated variable name 1: .std.resid
```

```r
lm1 |> glance() #key values eg R squared, can pull out
```

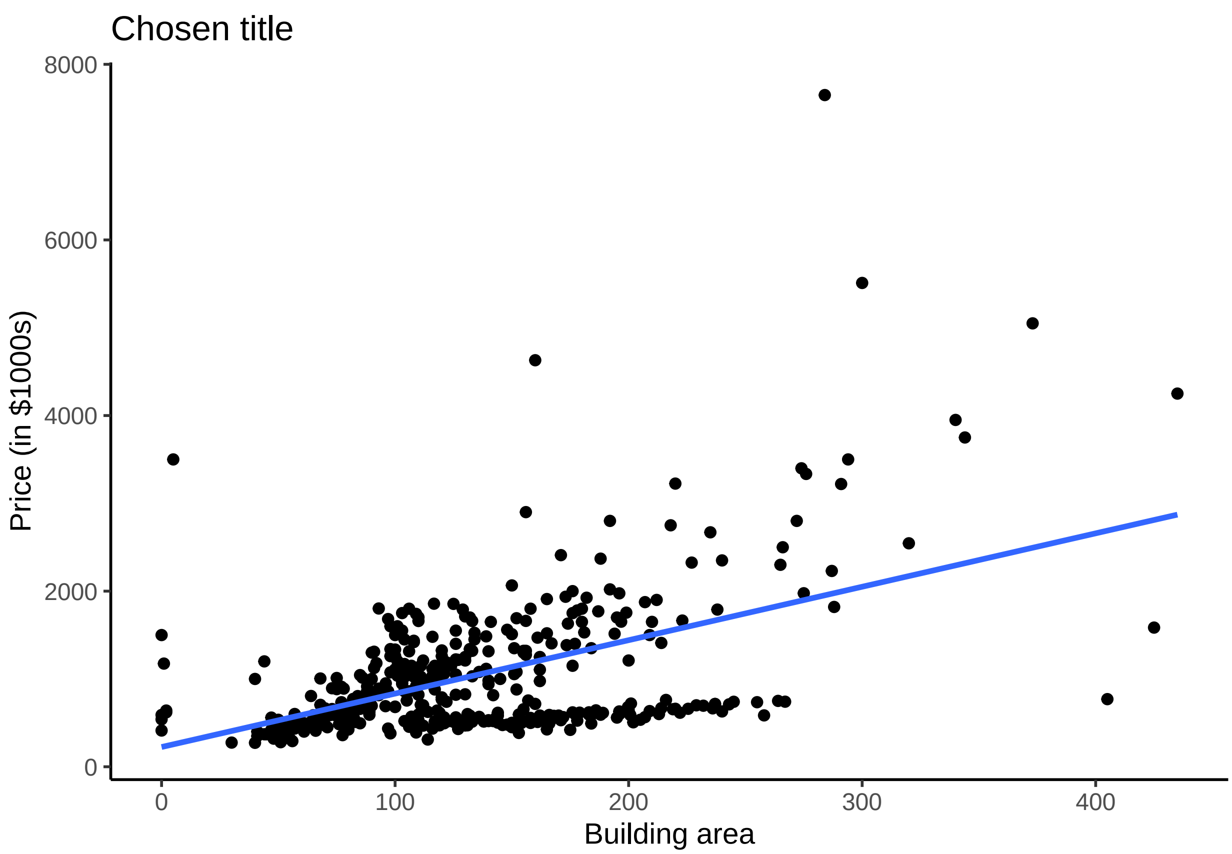
```
## # A tibble: 1 x 12
##   r.squared adj.r.squa~1 sigma stati~2  p.value    df logLik    AIC    BIC devia~3
```

```
##       <dbl>         <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.172         0.170  730.    86.7 7.41e-19     1 -3348. 6702. 6714.  2.22e8
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

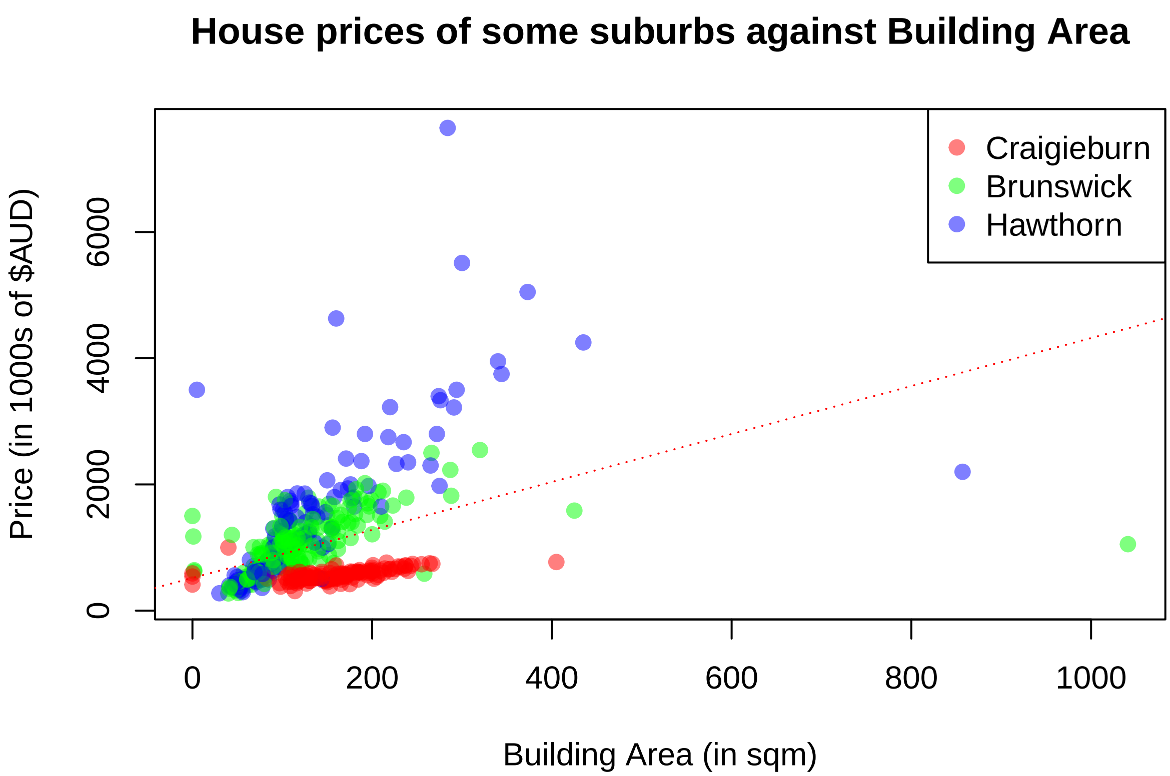
```
lm1 |> tidy() #conveniently puts summary into tibble format
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    518.      66.6        7.78 5.74e-14
## 2 BuildingArea     3.80     0.408      9.31 7.41e-19
```

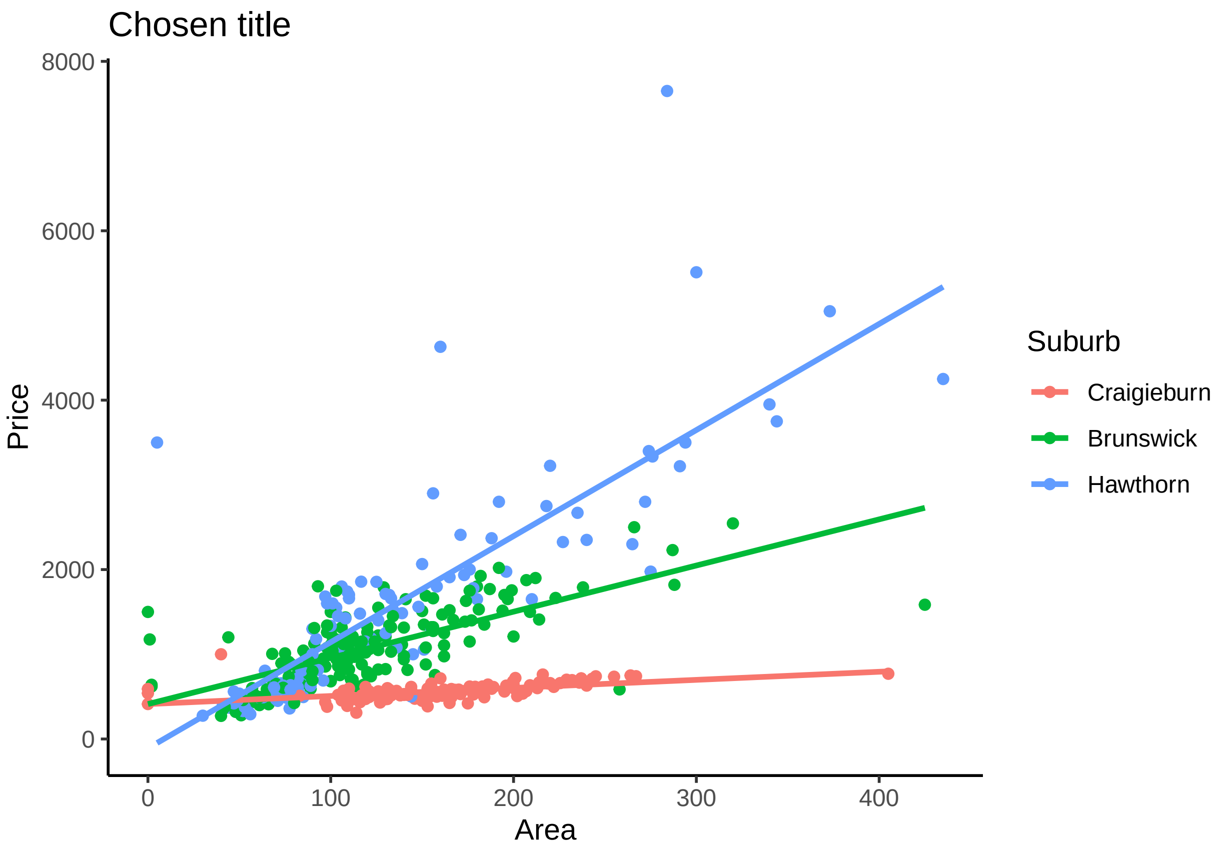
```
r2 <- lm1 |> glance() |> pull(r.squared)

melbdata.sub_out <- melbdata %>%
    filter(Suburb %in% c("Hawthorn", "Brunswick", "Craigieburn")) %>%
    mutate(Suburb = factor(Suburb, levels = c("Craigieburn", "Brunswick", "Hawthorn"))) %>%
    slice(-c(158,682)) #REMOVE THE WORST TWO OUTLIERS

lm1_alt <- lm(data = melbdata.sub_out, Price/1000 ~ BuildingArea)

#OUTLIERS HAVE BEEN REMOVED
ggplot(melbdata.sub_out |> select(BuildingArea, Price) |> drop_na()) +
    aes(x = BuildingArea, y = Price/1000) +
    geom_point() + geom_smooth(formula = y ~ x, method = "lm", se = FALSE) +
    theme_classic() + labs(x = "Building area", y = "Price (in $1000s)", title = "Chosen title")
```



```r
# Base R
lmfit1 <- lm(data = melbdata.sub, Price/1000 ~ BuildingArea)
#Get semi-transparent red green and blue
my.colours <- rgb(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), alpha = 0.5)
plot(Price/1000 ~ BuildingArea, data = melbdata.sub,
     main = "House prices of some suburbs against Building Area",
     xlab = "Building Area (in sqm)", ylab = "Price (in 1000s of $AUD)",
     col = my.colours[as.integer(melbdata.sub[["Suburb"]])],
     pch = 19)
legend("topright", legend = levels(melbdata.sub[["Suburb"]]),
       col = my.colours, pch = 19)
abline(lmfit1, col = "red", lty = "dotted")
```




```r
ggplot(melbdata.sub_out |> select(BuildingArea, Price, Suburb) |> drop_na()) +
    aes(x = BuildingArea, y = Price/1000, color = Suburb) +
    geom_point() +
    theme_classic() +
    labs(x = "Area", y = "Price", title = "Chosen title") +
    geom_smooth(formula = y ~ x, method = "lm", se = FALSE)
```



1.4 Finding association II

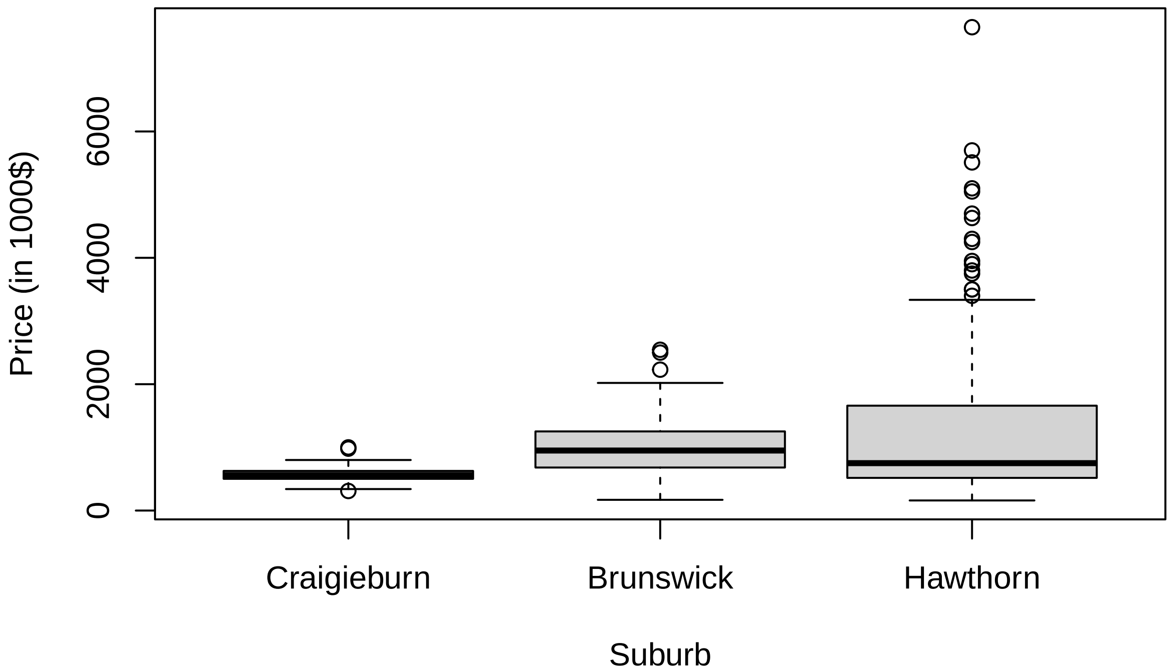
(a) Variability of house prices are complex and likely to be explained by many different factors. Construct a multiple linear regression here by examining if adding `Suburb` as a predictor will improve the prediction? Notice that `Suburb` is a categorical variable. Briefly describe how to interpret the regression coefficients returned by `lm`.

(b) There are many other variables in the data, you might consider whether adding the number of car spaces as a predictor improve the prediction model?

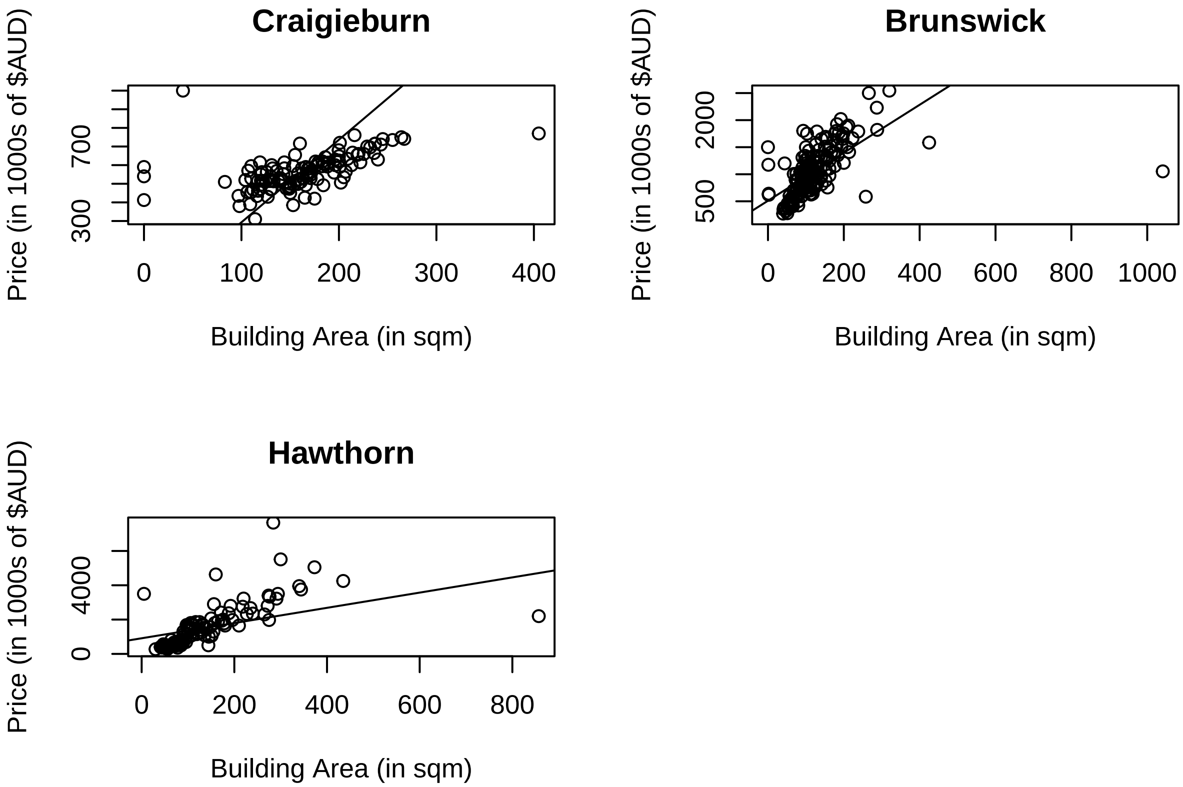
Solution

(a) Model fit below

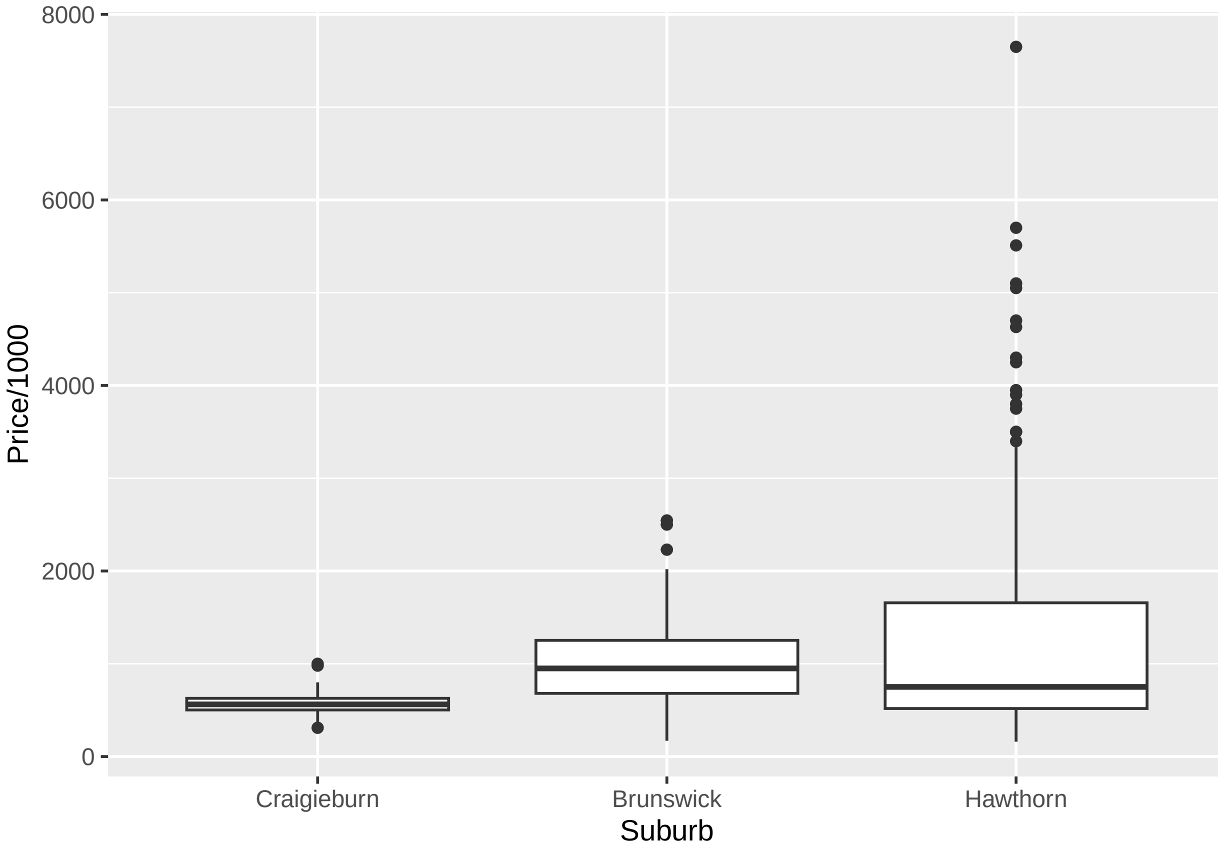
```
# Base R
boxplot(Price/1000 ~ Suburb, data = melbdata.sub, ylab = "Price (in 1000$)", xlab = "Suburb")
```



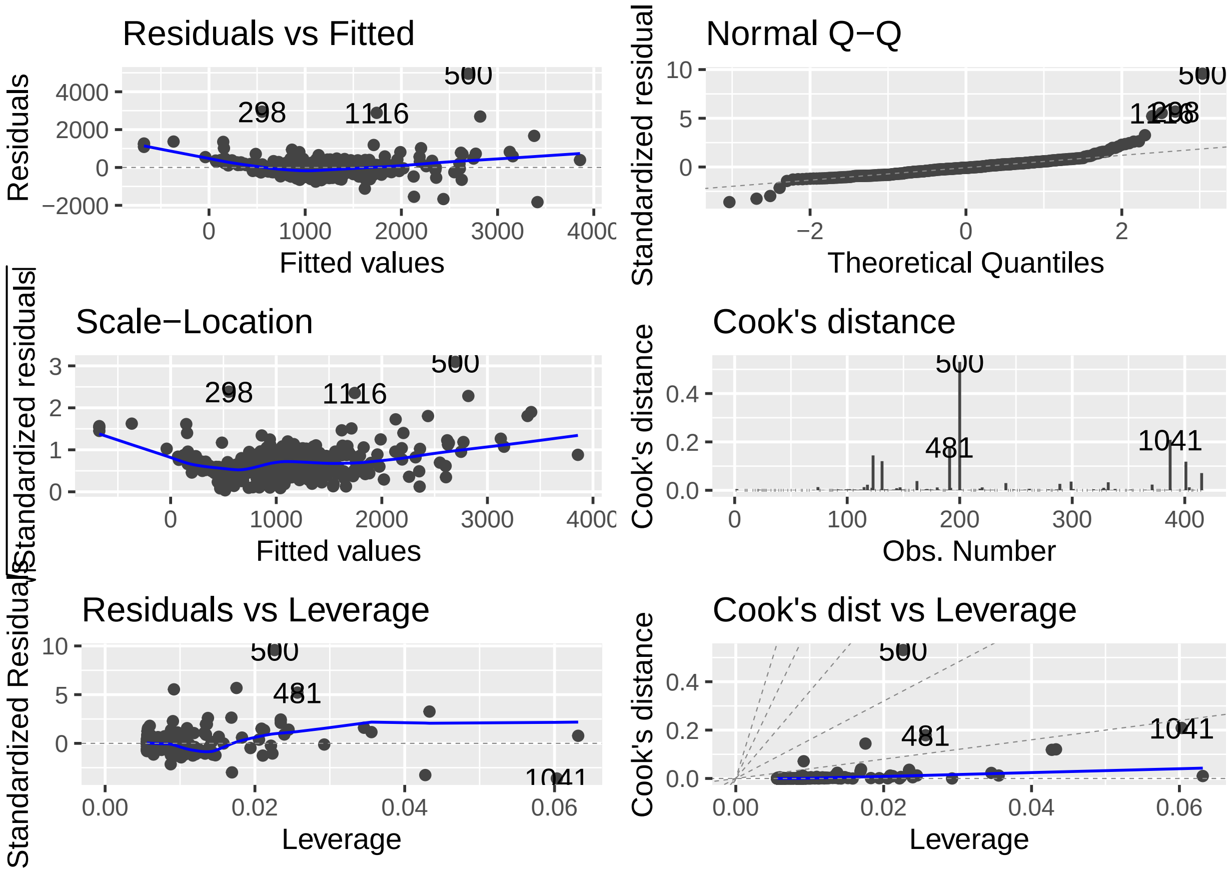

```
lm2 <- lm(Price/1000 ~ BuildingArea + Suburb, data = melb.data.sub)
coefs <- lm2 |> coef()

# Base R
par(mfrow = c(2, 2))
invisible(lapply(levels(melbdata.sub$Suburb), function(x) {
    plot(Price/1000 ~ BuildingArea, data = subset(melbdata.sub, Suburb == x), main = x,
         xlab = "Building Area (in sqm)", ylab = "Price (in 1000s of $AUD)")
    int <- coefs[1]
    if (any(adjust.ind <- grepl(paste0(x, "$"), names(coefs))))
        int <- int + coefs[adjust.ind]
    abline(int, coefs[2])
}))
```



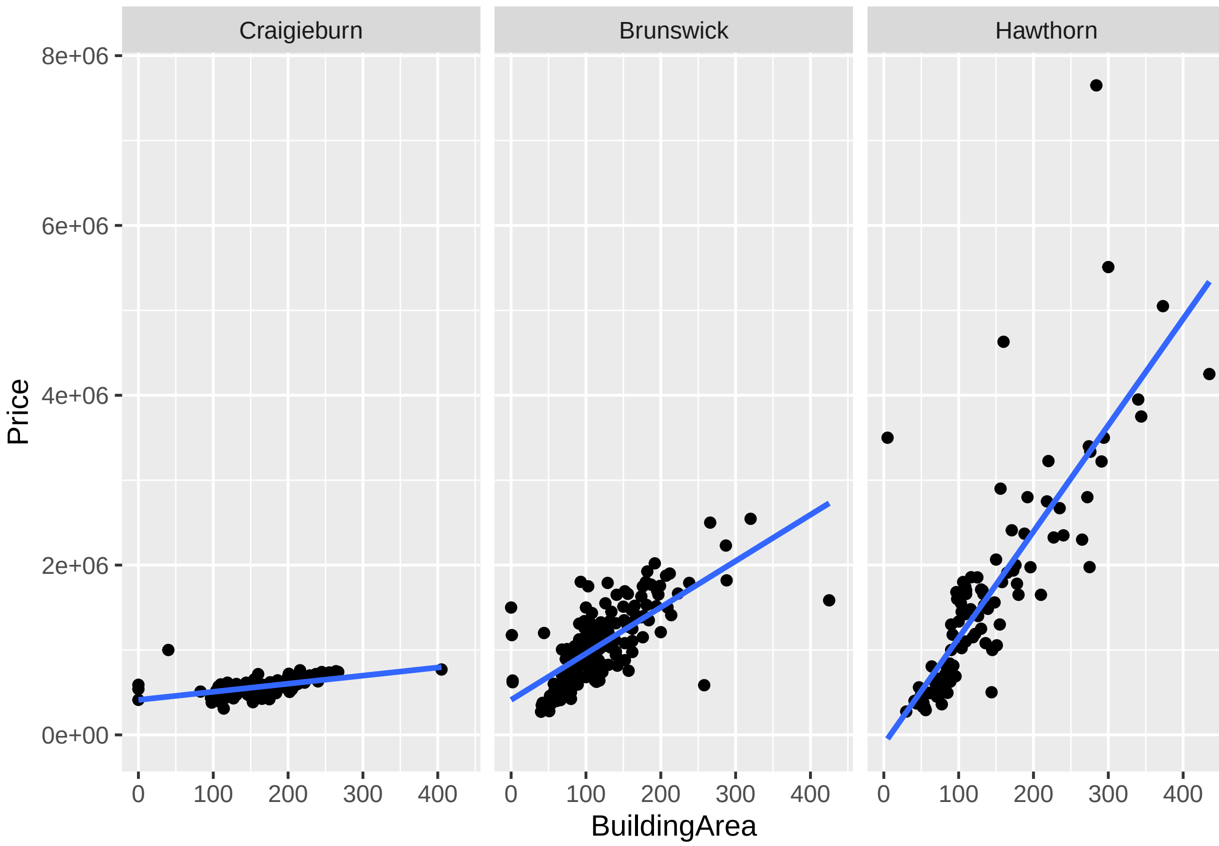

```r
# Tidyverse way
ggplot(melbdata.sub |> select(Suburb, Price) |> drop_na()) +
    aes(x = Suburb, y = Price/1000) + geom_boxplot()
```




```
lmfit2 <- lm(data = melbdata.sub %>%slice(-c(158,682)), Price/1000 ~ BuildingArea + Suburb)
coefs <- lmfit2$coefficients

autoplot(lmfit2, which = 1:6, nrow = 3, ncol = 2)
```




```
ggplot(melbdata.sub_out |> select(BuildingArea, Price, Suburb) |> drop_na()) +
    aes(x = BuildingArea, y = Price) +
    geom_point() +
    geom_smooth(formula = y ~ x, method = "lm", se = FALSE) +
    facet_wrap(~Suburb)
```




```
summary(lmfit2)
```

```
##
## Call:
## lm(formula = Price/1000 ~ BuildingArea + Suburb, data = melbdata.sub %>%
##     slice(-c(158, 682)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1829.6  -262.7   -41.5   185.2  4953.7
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -675.6983    80.1195  -8.434 5.71e-16 ***
## BuildingArea        7.6864     0.3991  19.260  < 2e-16 ***
## SuburbBrunswick   823.6223    63.9521  12.879  < 2e-16 ***
## SuburbHawthorn   1189.0488    69.2335  17.174  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 523.2 on 412 degrees of freedom
##   (709 observations deleted due to missingness)
## Multiple R-squared:  0.5768, Adjusted R-squared:  0.5737
## F-statistic: 187.2 on 3 and 412 DF,  p-value: < 2.2e-16
```

```
lmfit2 %>% glance()
```

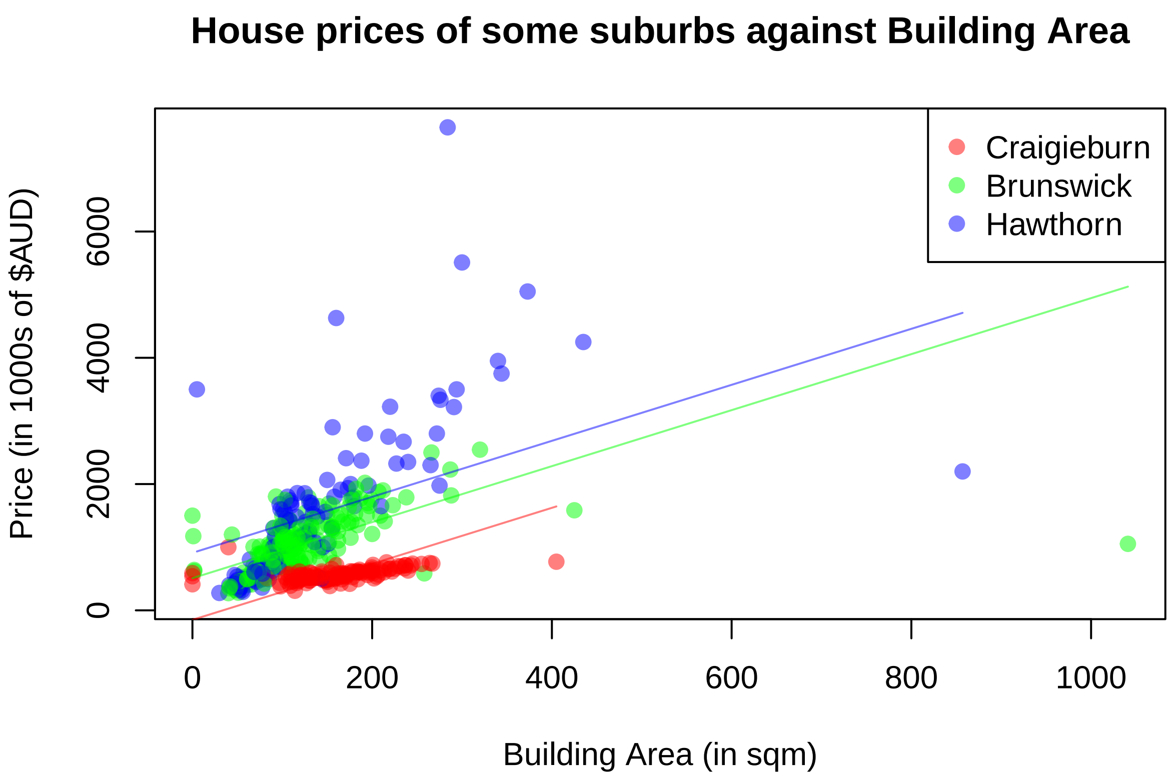
```
## # A tibble: 1 x 12
```

```
##   r.squared adj.r.squa~1 sigma stati~2  p.value    df logLik   AIC   BIC devia~3
##       <dbl>        <dbl> <dbl>   <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.577        0.574  523.    187. 1.47e-76     3 -3192. 6395. 6415.  1.13e8
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

```
r2s <- lmfit2 %>% glance() %>% pull(r.squared)
r2s <- summary(lmfit2)$r.squared
```

One way to highlight that the regression lines for the three suburbs are parallel is to put all three on the same graph, as follows.

```
# Base R
plot(Price/1000 ~ BuildingArea, data = melbdata.sub,
     main = "House prices of some suburbs against Building Area",
     xlab = "Building Area (in sqm)", ylab = "Price (in 1000s of $AUD)",
     col = my.colours[as.integer(melbdata.sub[["Suburb"]])],
     pch = 19)
legend("topright", legend = levels(melbdata.sub[["Suburb"]]),
       col = my.colours, pch = 19)
coefs <- coefficients(lmfit2)
names(my.colours) <- levels(melbdata.sub$Suburb)
r2 <- round(summary(lmfit1)$r.squared, 4)
obs.buildingarea.suburb <- subset(melbdata.sub, select = c("BuildingArea", "Suburb"))
obs.buildingarea.suburb <- na.omit(obs.buildingarea.suburb)
buildingarea.by.suburb <- with(obs.buildingarea.suburb, split(BuildingArea, Suburb))
buildingarea.by.suburb <- lapply(buildingarea.by.suburb, range)

lapply(levels(melbdata.sub$Suburb), function(x) {
    pred.df <- data.frame(BuildingArea = buildingarea.by.suburb[[x]],
                          Suburb = x)
    lines(pred.df[["BuildingArea"]], predict(lm2, newdata = pred.df),
          col = my.colours[which(levels(obs.buildingarea.suburb[["Suburb"]]) == x)])
})
```



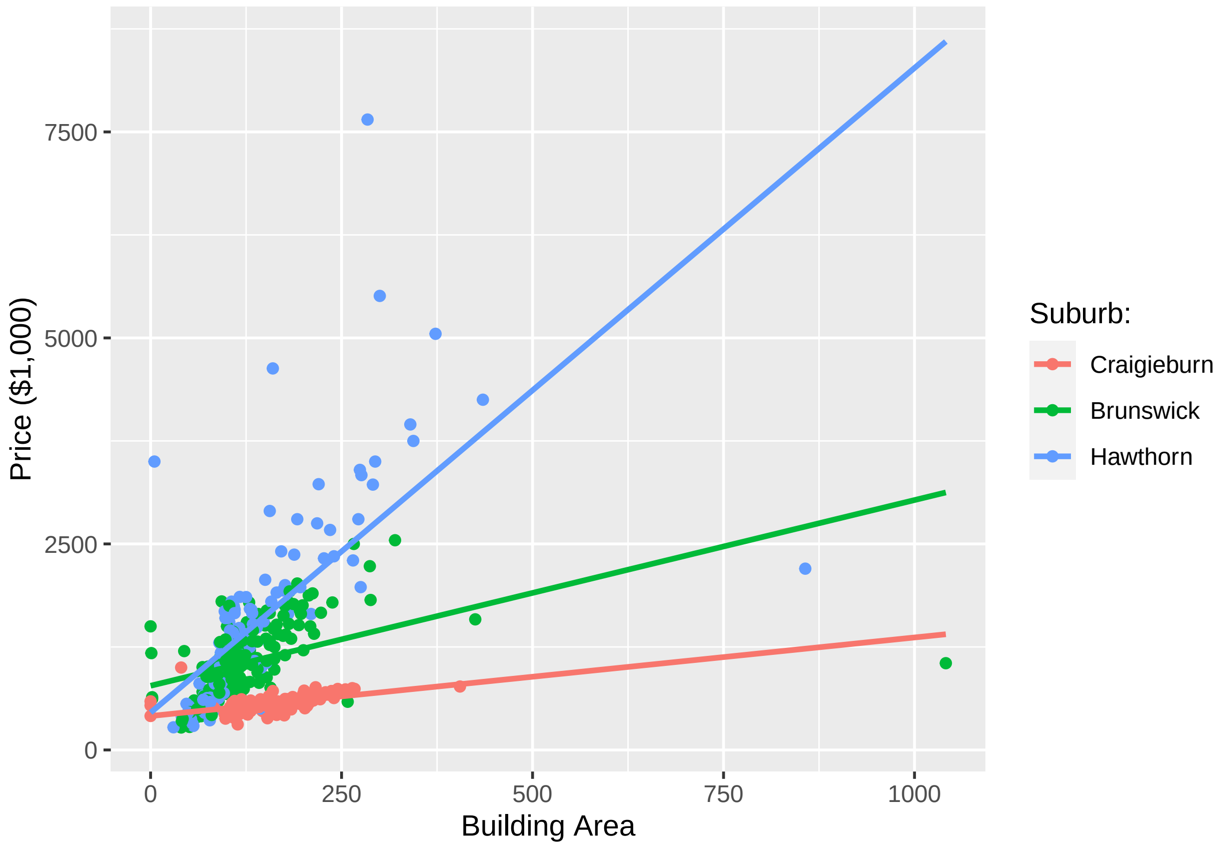

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

1.4.1 Embedding model fit in ggplot

`ggplot` includes a really clever trick to support easily constructing line fits (smoothed or linear or ...) however the interactions between the model specification and the plotting specification can be subtle, resulting in graphs that do not match the numerical analysis, which are then at best misleading.

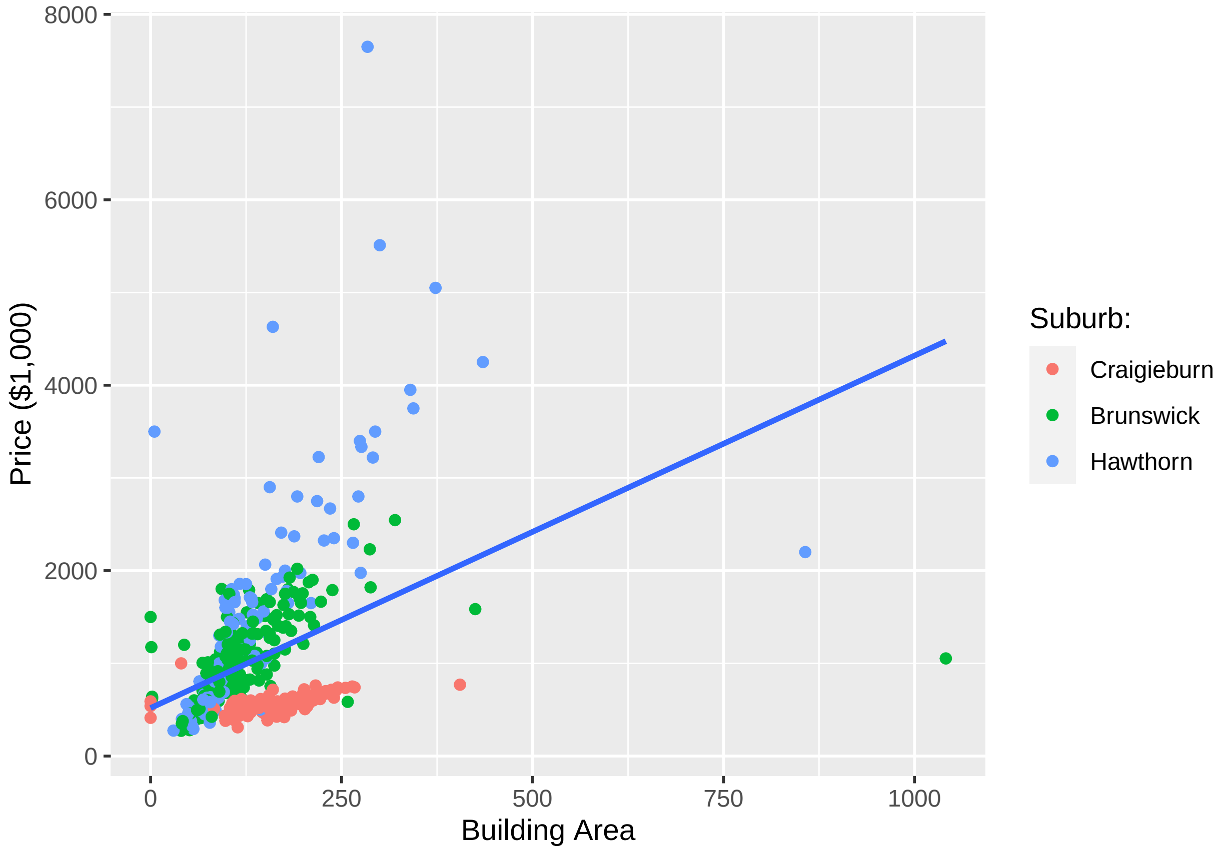
An example of fitting different models.

```
#
library(ggplot2)
ggplot(melbdata.sub, aes(x = BuildingArea, y = Price/1000, color=Suburb)) +
  geom_point(na.rm = TRUE) +
  geom_smooth(formula = "y~x", method = "lm", se = FALSE, fullrange = TRUE, na.rm = TRUE) +
  xlab("Building Area") + ylab("Price ($1,000)") + labs(colour = "Suburb:")
```



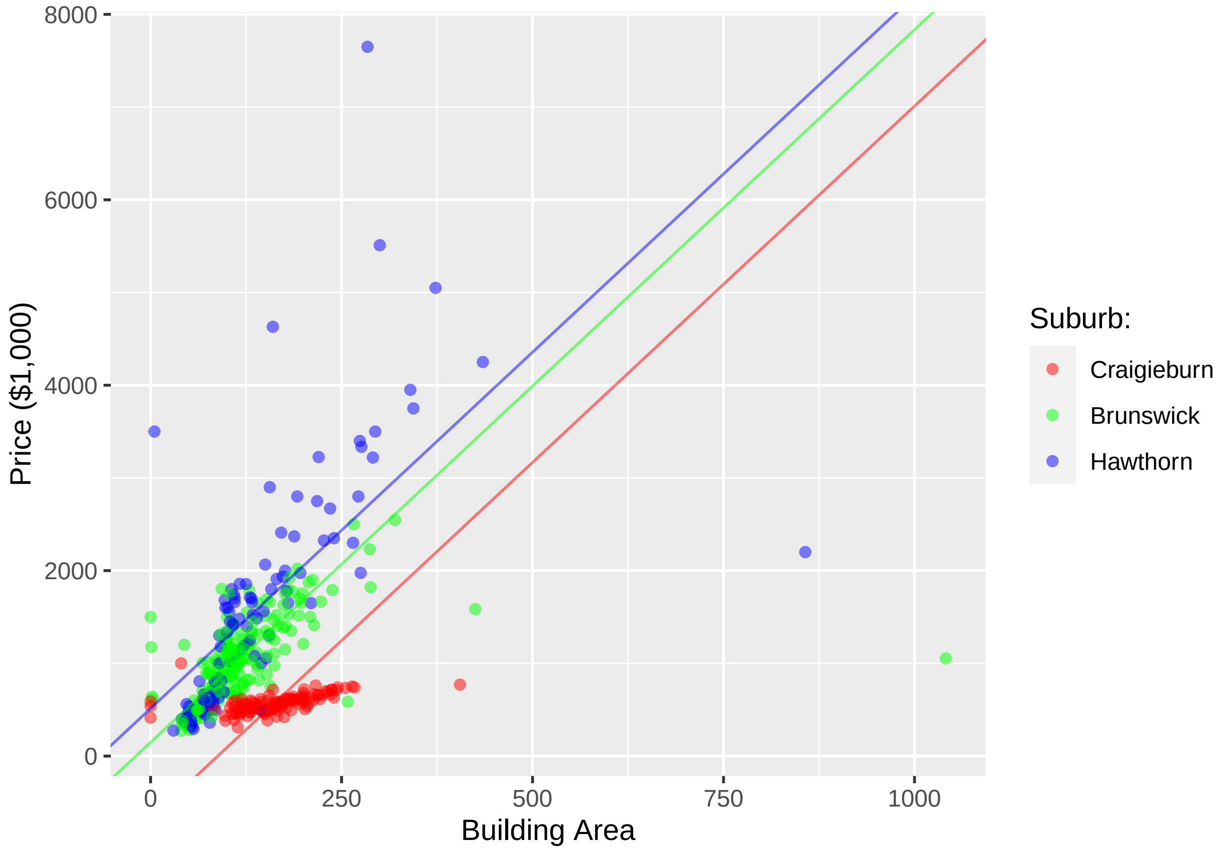
Specifying the colour slightly differently, results in a different model!

```
ggplot(melbdata.sub, aes(x = BuildingArea, y = Price/1000)) +
  geom_point(aes(color=Suburb), na.rm = TRUE) +
  geom_smooth(formula="y~x", method = "lm", se = FALSE, fullrange=TRUE, na.rm = TRUE) +
  xlab("Building Area") + ylab("Price ($1,000)") + labs(colour="Suburb:")
```



A simple way to avoid the problem is to make sure that the model being plotted is the original model used for the numerical analysis:

```
# reuse previous color key
colScale <- scale_colour_manual(name = "Suburb:", values = my.colours) # name is used as legend title
ggplot(melbdata.sub, aes(x = BuildingArea, y = Price/1000, col = Suburb) ) +
  geom_point(na.rm = TRUE) +
  xlab("Building Area") + ylab("Price ($1,000)") + colScale +
  sapply(unique(melbdata.sub$Suburb), function(x) {
       int <- coefs[1]
       if (any(adjust.ind <- grepl(paste0(x, "$"), names(coefs))))
           int <- int + coefs[adjust.ind]
       geom_abline( intercept=int, slope=coefs[2], col=my.colours[x] )
    })
```



1.4.2 Commentary on Models

The `Suburb` predictor improves the fit of the model by increasing the R^2 from 0.1725 to 0.5767645. However, the adjusted R^2 is amore appropriate goodness of fit measure when there is more than one predictor in the model since adding another predictor will always increase the R^2. In this case the adjusted R^2 increases by a similar amount suggesting Suburb is a good additional predictor.

Interpreting the `BuildingArea` slope has the interpretation that for each unit increase in square meter of building size, the expected average price would increase by $4435. Interpreting the categorical predictors needs to be done by intercept adjustment. The first categorical level of `Suburb` (Brunswick) becomes the baseline intercept and the other suburbs are adjusted against the baseline intercept. In this case Craigieburn and Hawthorn have adjustments of -660,000 and 400,000 respectively. This should be interpretted that properties in Craigieburn are $660,000 cheaper than Brunswick on average (if BuildingArea is held fixed). Hawthorn properties are $400,000 more expensive than Brunswick. This is consistent with the graphical summary in the boxplot which indicates without adjusting for Building Area, Craigie burn tends to have cheaper houses with low variance while Hawthorn has a large variance in house prices with many very expensive outlying properties.

(b) Adding the number of car spaces in to the model and compare the goodness of fit measures.

```
lmfit4 <- lm(data = melbdata.sub, Price ~ BuildingArea + Suburb + Car)
summary(lmfit4)
```

```
##
## Call:
## lm(formula = Price ~ BuildingArea + Suburb + Car, data = melbdata.sub)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -3417281  -273352   -59191   252474  5001704
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -447547.1    89421.2  -5.005 8.37e-07 ***
## BuildingArea       3761.7      351.3  10.708  < 2e-16 ***
## SuburbBrunswick  781107.7    73776.4  10.588  < 2e-16 ***
## SuburbHawthorn  1144544.5    78263.7  14.624  < 2e-16 ***
## Car              220740.5    34616.8   6.377 4.98e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 588500 on 402 degrees of freedom
##   (720 observations deleted due to missingness)
## Multiple R-squared:  0.477,  Adjusted R-squared:  0.4718
## F-statistic: 91.66 on 4 and 402 DF,  p-value: < 2.2e-16
```

Adding car spaces seems to improve the prediction model if `BuildingArea` and `Suburb` are already included in the model. The goodness of fit metrics (both raw and adjusted) increase.

1.5 Impact of outliers

Model construction can be affected by unwanted variation and noise such as outliers. For example, houses with very small building areas of 5sqm and lower and larger places over 300 sqm look like outliers. How would you assess the impact of outliers?

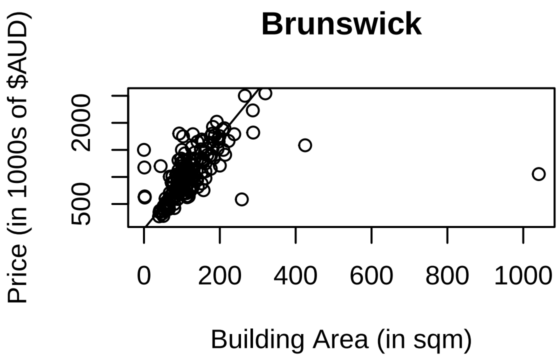
Solution

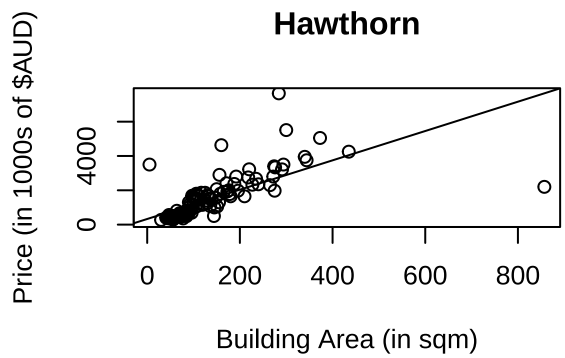
A simple strategy to assess the impact of outliers is to remove the outliers and see if you can improve the prediction model.

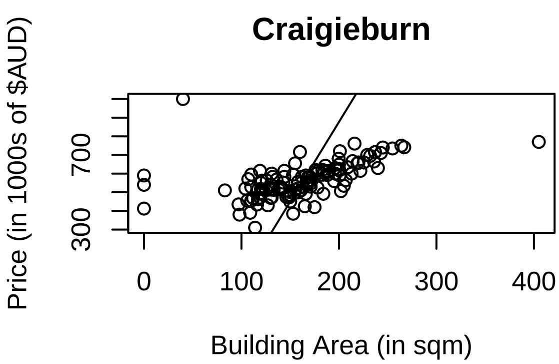
```r
melbdata.sub.2 <- subset(melbdata.sub, BuildingArea > 10 & BuildingArea < 300)
lmfit3 <- lm(data = melbdata.sub.2, Price/1000 ~ BuildingArea + Suburb)
coefs <- lmfit3$coefficients
par(mfrow = c(2, 2))
invisible(lapply(unique(melbdata.sub$Suburb), function(x) {
    plot(Price/1000 ~ BuildingArea, data = subset(melbdata.sub, Suburb == x), main = x,
         xlab = "Building Area (in sqm)", ylab = "Price (in 1000s of $AUD)")
    int <- coefs[1]
    if (any(adjust.ind <- grepl(paste0(x, "$"), names(coefs))))
        int <- int + coefs[adjust.ind]
    abline(int, coefs[2])
}))
summary(lmfit3)
```

```
##
## Call:
## lm(formula = Price/1000 ~ BuildingArea + Suburb, data = melbdata.sub.2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1658.0  -260.8   -30.5   192.6  4895.4
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)     -832.4033    80.4592  -10.35   <2e-16 ***
## BuildingArea       8.5448     0.4234   20.18   <2e-16 ***
## SuburbBrunswick  870.8069    56.3822   15.45   <2e-16 ***
## SuburbHawthorn  1160.2793    61.6807   18.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 442.6 on 396 degrees of freedom
##   (81 observations deleted due to missingness)
## Multiple R-squared:  0.5984, Adjusted R-squared:  0.5954
## F-statistic: 196.7 on 3 and 396 DF,  p-value: < 2.2e-16
```







Removing the outliers does improve the fit of the model as the overall residual standard error has decreased and the goodness of fit metrics have increased to around 0.6. Also visually we can see an improved fit for Hawthorn and Brunswick. Although there still is a poor fit for Cragieburn

1.6 Prediction

Predict the price of a house in Hawthorn with 2 car spaces and 100 sqm in building area. What is the 95% confidence interval of your prediction value?

Solution

```
predict(lmfit4, data.frame(Suburb = "Hawthorn", BuildingArea = 100, Car = 2), interval = "confidence")
```

```
##       fit     lwr     upr
## 1 1514653 1393868 1635439
```

If using the model in question 1.4, then the predicted value for an average property with those features would be $1,514,653.