MATH3714/5714M Practical Solutions

Jochen Voss

2023-12-14

This document has comments, solutions and marking criteria for the practical of the module MATH3714/5714M. I would expect submitted solutions to be shorter, and to have more discussion/explanations but less R code than these notes.

Task 1 (6 marks)

We start our analysis by loading the training data set into R:

```
train <- read.csv("https://www1.maths.leeds.ac.uk/~voss/data/housing/train.csv")
```

Model Fitting

Fitting a model which describes `medianHouseValue` as a function of `medianIncome` is straightforward:

```
m <- lm(medianHouseValue ~ medianIncome, data = train)
summary(m)
```

```
##
## Call:
## lm(formula = medianHouseValue ~ medianIncome, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -515117  -51344  -13978   36193  370777
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43720.2     1919.1   22.78   <2e-16 ***
## medianIncome  40179.5      482.1   83.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74180 on 9836 degrees of freedom
## Multiple R-squared:  0.4139, Adjusted R-squared:  0.4139
## F-statistic:  6947 on 1 and 9836 DF,  p-value: < 2.2e-16
```

The resulting model is

$$y = 43{,}720.2 + 40{,}179.5\,x + \varepsilon,$$

where x is the median income for each area, y is the median house price for the area, and $\varepsilon \sim \mathcal{N}(0, 74{,}180^2)$.

Regression Diagnostics

Regression diagnostics are discussed in sections 9 and 11 of the lecture notes. Here are some measures we have discussed (others are possible, too).

Coefficient of Multiple Determination

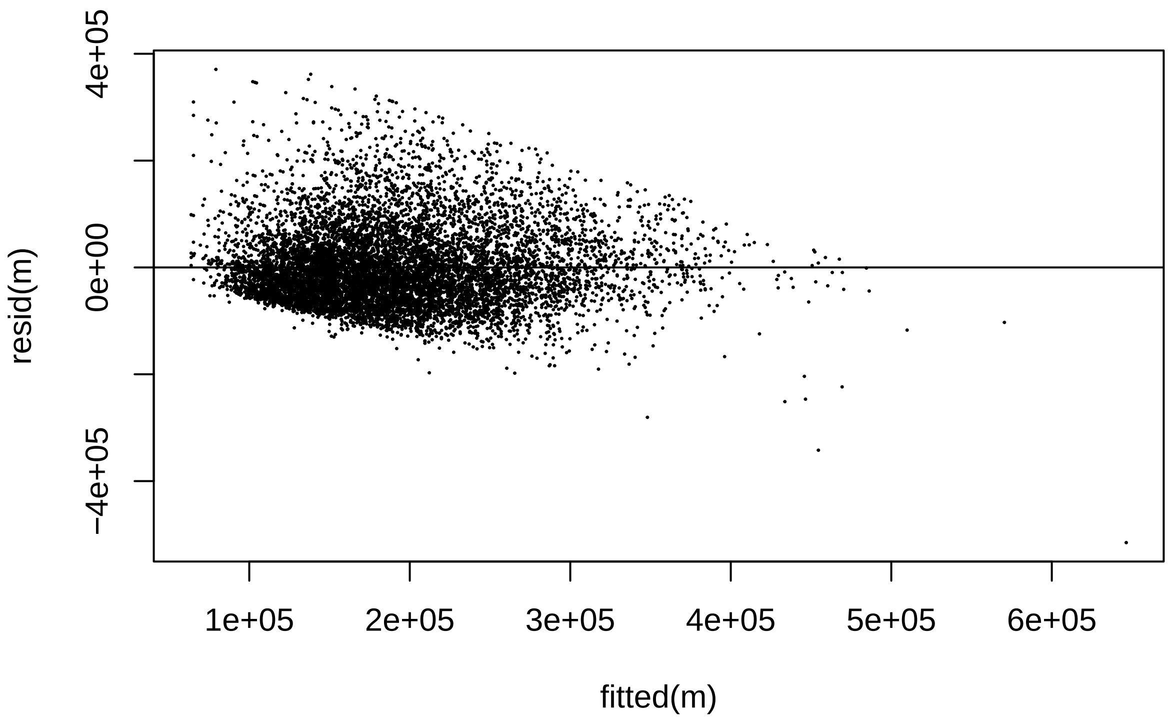
The R^2 value states how much of the fluctuations in the output can be explained by the inputs, as a ratio between 0 and 1. Larger values indicate better model fit.

From the summary output above we see that our model has $R^2 = 0.4139$. This means that the variance of the residuals is nearly 60% of the variance of the median house prices. Hopefully a better model can be found in task 2.

Residual plots

This is a plot of the residuals against the fitted values:

```r
plot(fitted(m), resid(m), cex = .1)
abline(h = 0)
```

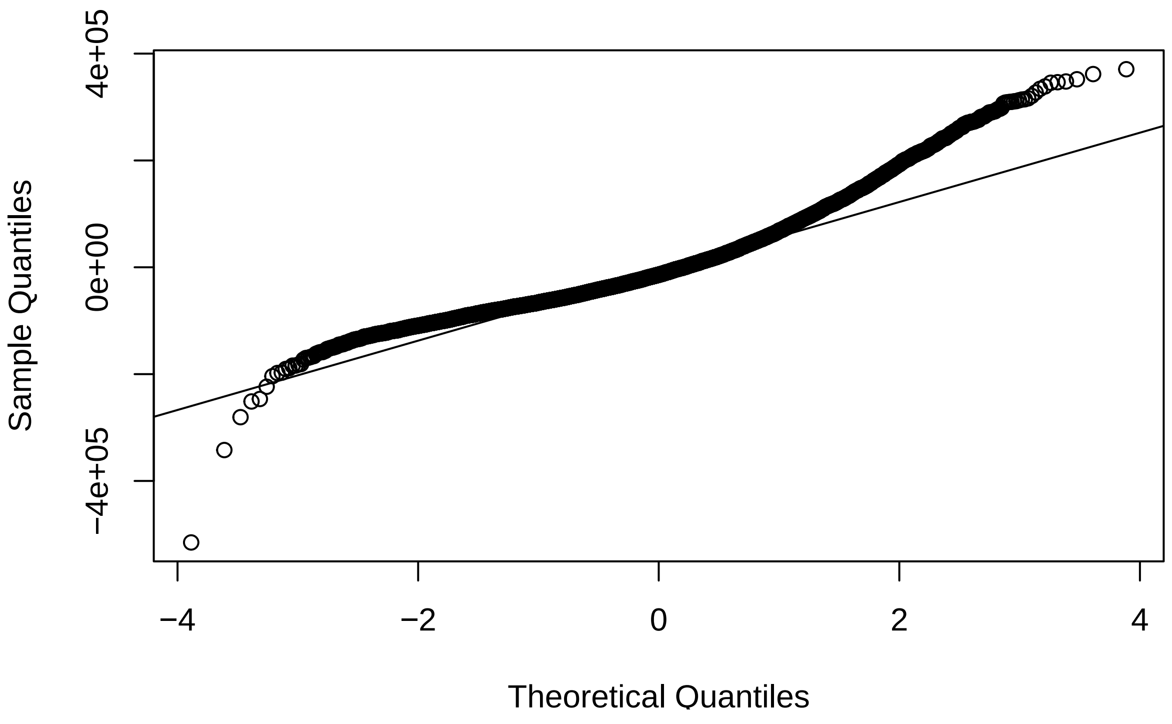


If the model fits well, the residuals in this plot should form a horizontal band of constant width, centred around 0. Here it seems that the variance of the residuals around $\hat{y} = 20{,}000$ might be larger than near the boundaries, and possibly the mean of the residuals is negative for small fitted values. From the plot, model fit is not very good.

Q-Q plot

A Q-Q plot compares the empirical quantiles of the residuals to the quantiles of the fitted normal distribution. A straight line indicates good model fit.

```r
qqnorm(resid(m), main = NULL)
qqline(resid(m))
```



From the plot it seems clear that the residuals are not normally distributed.

Confidence Interval

Methods for computing confidence intervals are discussed in section 7.1 of the lecture notes. Here we use the `summary(m)` output to get all required information, but different ways of computing confidence intervals can of course also be used.

```
summary(m)
```

```
##
## Call:
## lm(formula = medianHouseValue ~ medianIncome, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -515117  -51344  -13978   36193  370777
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43720.2     1919.1   22.78   <2e-16 ***
## medianIncome  40179.5      482.1   83.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74180 on 9836 degrees of freedom
## Multiple R-squared:  0.4139, Adjusted R-squared:  0.4139
## F-statistic:  6947 on 1 and 9836 DF,  p-value: < 2.2e-16
```

```
alpha <- 0.05
n <- nrow(train)
p <- 1
t <- qt(1 - alpha/2, n - p - 1)
```

```
cat("[", 43720.2 - 1919.1*t, ", ", 43720.2 + 1919.1*t, "]\n", sep = "")
```

```
## [39958.37, 47482.03]
```

Marking Criteria

- The data has been loaded correctly.
- The resulting model is correct and explicitly stated (including the error variance).
- There is some discussion of the model fit, including at least one of the measures discussed in the lecture notes.
- The confidence interval is computed correctly.

Task 2 (12 marks)

This task is very open-ended, and I expect that different students will come up with different solutions. Here I will just give some ideas and examples.

Baseline Model

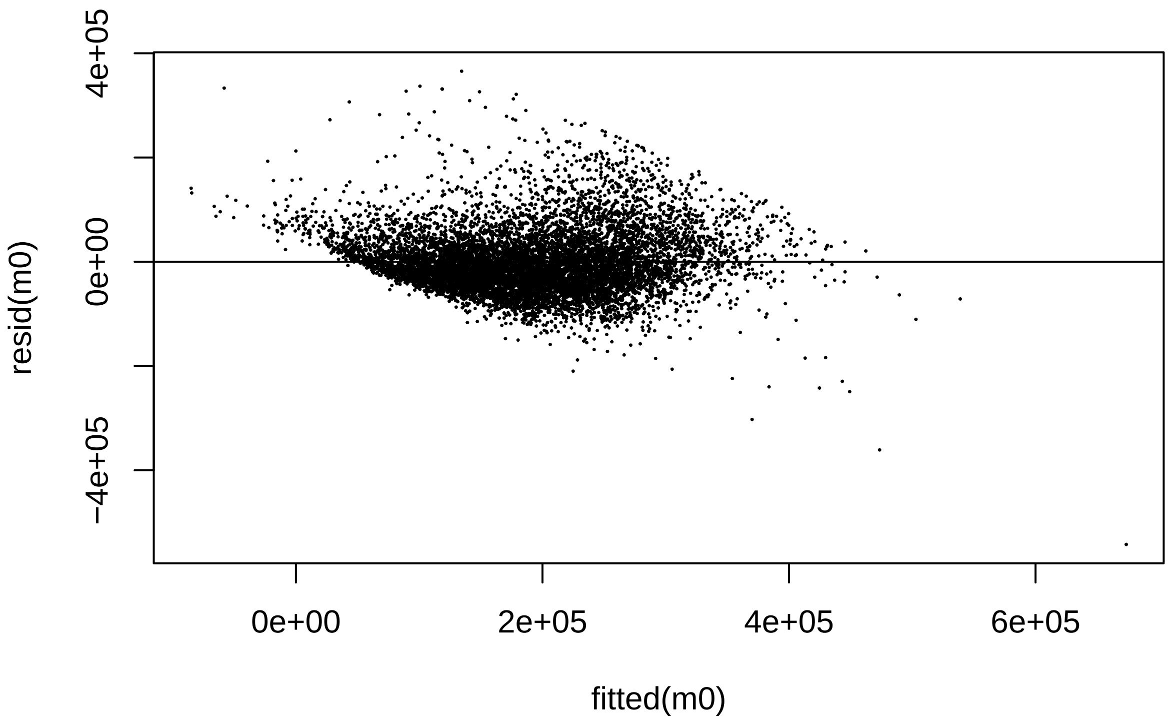
As a baseline, we fit the model which uses all input variables, without any transformations or interaction terms:

```
m0 <- lm(medianHouseValue ~ ., data = train)
summary(m0)
```

```
##
## Call:
## lm(formula = medianHouseValue ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -542334  -38926   -8820   29096  365582
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -3.345e+06  7.877e+04 -42.467  < 2e-16 ***
## longitude         -3.981e+04  8.983e+02 -44.312  < 2e-16 ***
## latitude          -3.934e+04  8.438e+02 -46.616  < 2e-16 ***
## housingMedianAge   9.895e+02  5.606e+01  17.650  < 2e-16 ***
## totalRooms        -7.969e+00  1.030e+00  -7.740 1.09e-14 ***
## totalBedrooms      8.901e+01  9.238e+00   9.635  < 2e-16 ***
## population        -3.693e+01  1.527e+00 -24.178  < 2e-16 ***
## households         6.672e+01  9.984e+00   6.683 2.47e-11 ***
## medianIncome       3.930e+04  5.332e+02  73.699  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60790 on 9829 degrees of freedom
## Multiple R-squared:  0.6067, Adjusted R-squared:  0.6064
## F-statistic:  1895 on 8 and 9829 DF,  p-value: < 2.2e-16
```

The adjusted R^2 value is 0.6064, which is a significant improvement from the model in task 1 (where we got 0.4139). We also check the residual plot for this model:

```r
plot(fitted(m0), resid(m0), cex = .1)
abline(h = 0)
```



There is some curvature visible in the plot (residuals in the middle of the range seem to be lower). We will try to find a better model to improve on this.

Transformation of Variables

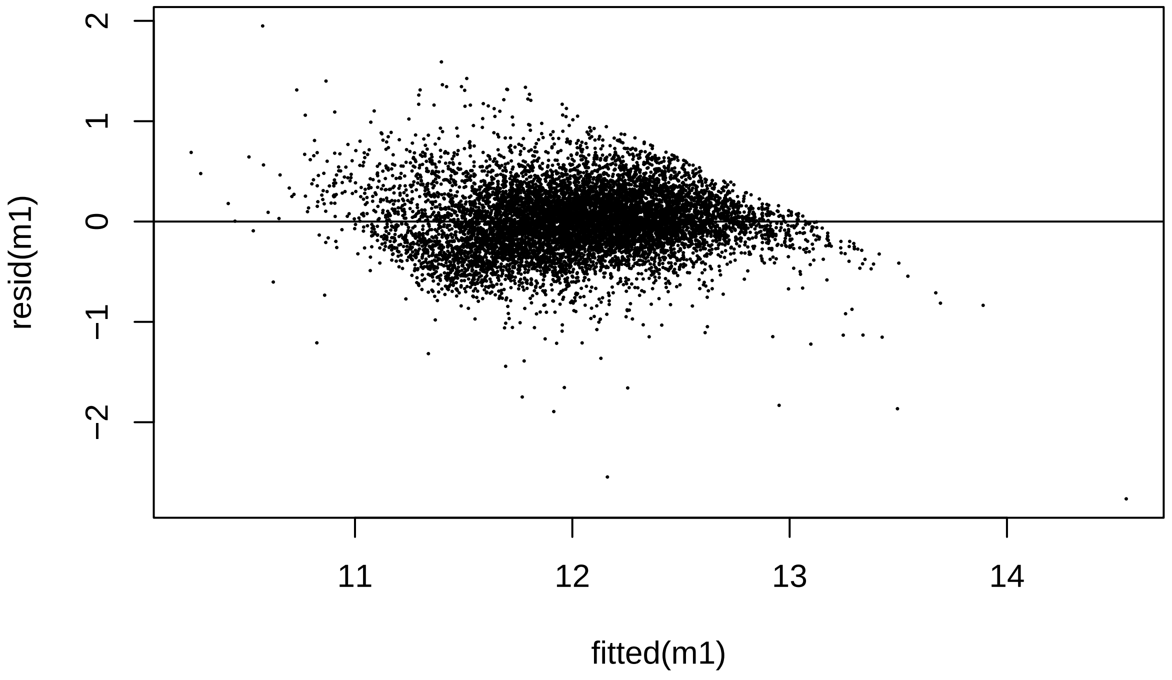
Logarithmic Transformation of the response

The residual plot above seemed to indicate a non-linear relationship between inputs and outputs. One straightforward way to improve the model is to transform the response variable. I tried `log()` and `sqrt()`. Since `log()` seemed to work better, I will use this transformation here.

```r
m1 <- lm(log(medianHouseValue) ~ ., data = train)
summary(m1)
```

```
##
## Call:
## lm(formula = log(medianHouseValue) ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.76375 -0.19588 -0.00362  0.18522  1.94922
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.053e+01  4.148e-01 -25.382  < 2e-16 ***
## longitude        -2.591e-01  4.730e-03 -54.789  < 2e-16 ***
## latitude         -2.619e-01  4.443e-03 -58.954  < 2e-16 ***
## housingMedianAge  3.597e-03  2.952e-04  12.187  < 2e-16 ***
## totalRooms       -5.126e-05  5.421e-06  -9.455  < 2e-16 ***
## totalBedrooms     5.163e-04  4.864e-05  10.614  < 2e-16 ***
## population       -1.770e-04  8.041e-06 -22.007  < 2e-16 ***
## households        3.138e-04  5.257e-05   5.969 2.48e-09 ***
## medianIncome      2.078e-01  2.808e-03  74.020  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3201 on 9829 degrees of freedom
## Multiple R-squared:  0.6382, Adjusted R-squared:  0.6379
## F-statistic:  2167 on 8 and 9829 DF,  p-value: < 2.2e-16
```

```r
plot(fitted(m1), resid(m1), cex = .1)
abline(h = 0)
```



Transformations of the input variables and Variable Selection

Ideas:

- The number of rooms and bedrooms per household/person might be more relevant than the total number of rooms and bedrooms.
- The number of persons per household might be more relevant than the total number of persons and/or households.
- Each of the variables might individually be transformed, using functions like `log()`, `sqrt()` or `(...)^2`.
- We can use the `regsubsets()` function from the `leaps` package to find a good subset of variables.

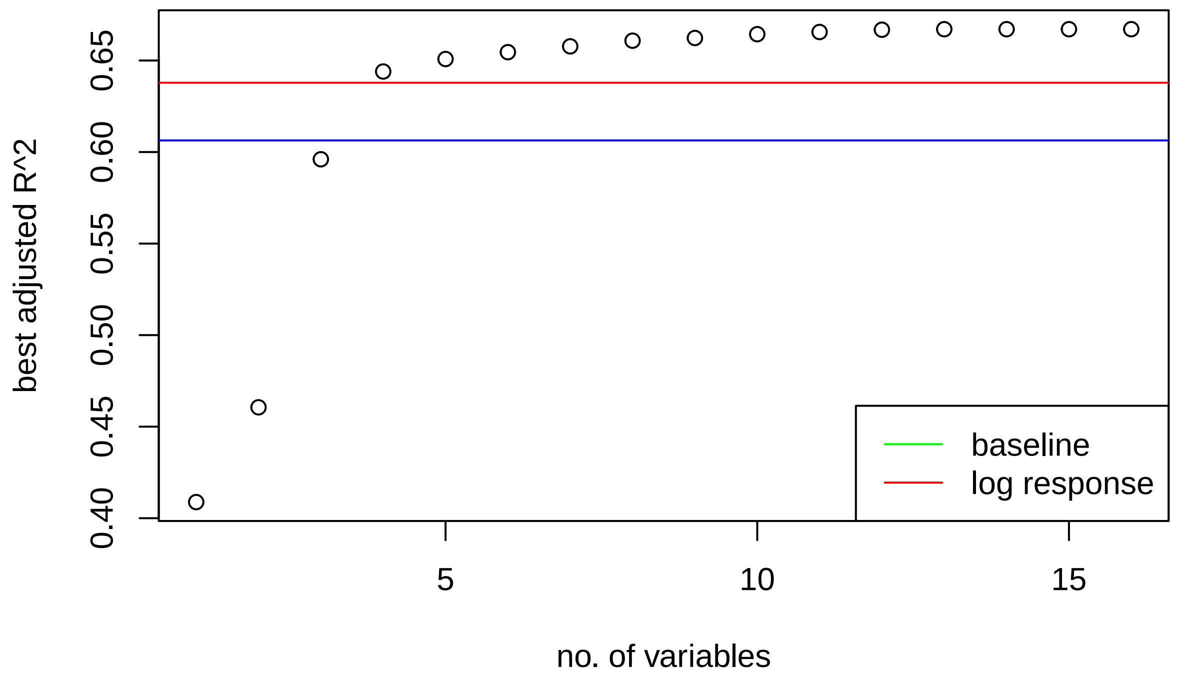
```r
library(leaps)
models <- regsubsets(log(medianHouseValue) ~
                       longitude +
                       latitude +
                       housingMedianAge +
                       totalRooms +
                       totalBedrooms +
                       I(totalRooms/population) +
                       I(totalBedrooms/population) +
                       I(totalRooms/households) +
                       I(totalBedrooms/households) +
                       I(population/totalRooms) +
                       I(population/totalBedrooms) +
                       I(households/totalRooms) +
                       I(households/totalBedrooms) +
                       I(log(totalRooms/population)) +
                       I(log(totalBedrooms/population)) +
                       I(log(totalRooms/households)) +
                       I(log(totalBedrooms/households)) +
                       households +
                       I(households/population) +
                       I(population/households) +
                       I(log(population/households)) +
                       population +
                       medianIncome,
                     data = train, nvmax=15, method = "exhaustive")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 2 linear dependencies found
```

```
## Reordering variables and trying again:
```

I plot the adjusted R^2 values for the best model with each number of variables. For comparison, I also plot the adjusted R^2 values for the baseline model (blue line) and the model with the transformed response (red line).

```
plot(summary(models)$adjr2,
     xlab = "no. of variables", ylab = "best adjusted R^2")
abline(h = summary(m0)$adj.r.squared, col = "blue")
abline(h = summary(m1)$adj.r.squared, col = "red")
legend("bottomright", legend = c("baseline", "log response"),
       col = c("green", "red"), lty = 1)
```



We can see that only four variables are needed to get a model which is better than the the two models we have seen so far. Here I select two of models the models found by `regsubsets`.

```
t(summary(models)$which[c(4,8),])
```

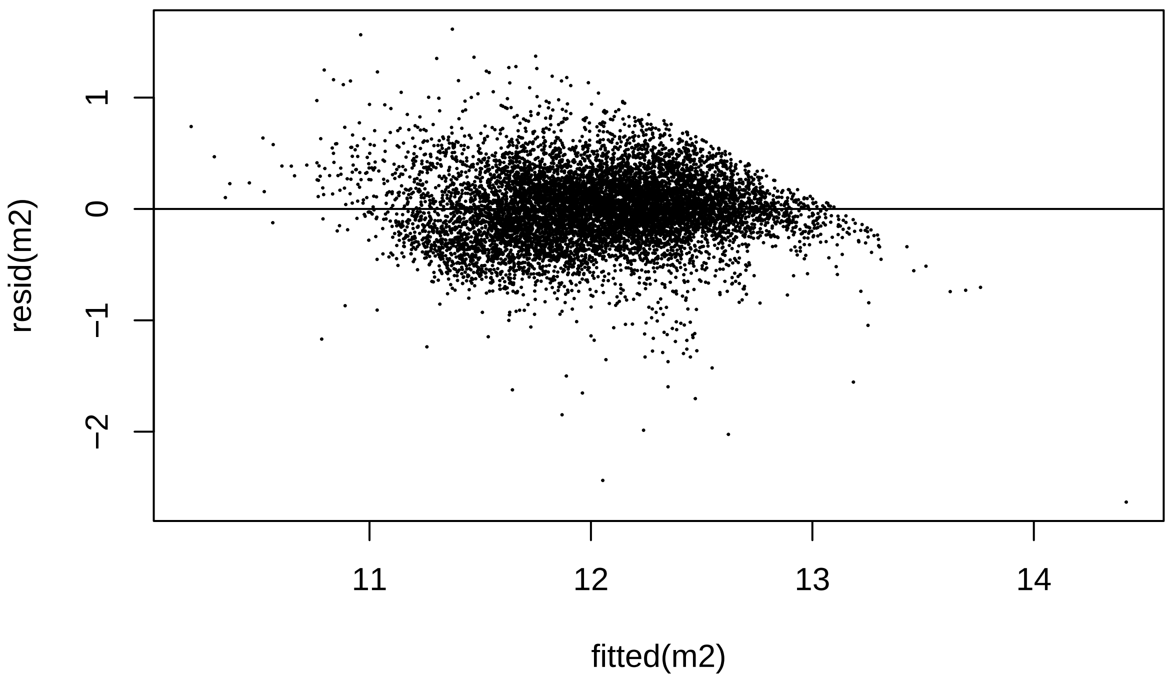
```
##                                      4     8
## (Intercept)                       TRUE  TRUE
## longitude                         TRUE  TRUE
## latitude                          TRUE  TRUE
## housingMedianAge                 FALSE  TRUE
## totalRooms                       FALSE FALSE
## totalBedrooms                    FALSE FALSE
## I(totalRooms/population)         FALSE FALSE
## I(totalBedrooms/population)      FALSE FALSE
## I(totalRooms/households)         FALSE FALSE
## I(totalBedrooms/households)      FALSE FALSE
## I(population/totalRooms)         FALSE FALSE
## I(population/totalBedrooms)      FALSE FALSE
## I(households/totalRooms)         FALSE  TRUE
## I(households/totalBedrooms)      FALSE FALSE
## I(log(totalRooms/population))    FALSE FALSE
## I(log(totalBedrooms/population)) FALSE  TRUE
## I(log(totalRooms/households))    FALSE FALSE
## I(log(totalBedrooms/households)) FALSE FALSE
## households                       FALSE  TRUE
## I(households/population)          TRUE FALSE
## I(population/households)         FALSE  TRUE
## I(log(population/households))    FALSE FALSE
## population                       FALSE FALSE
## medianIncome                      TRUE  TRUE
```

Model `m2` is the smallest model which beats `m1` in terms of the adjusted R^2 value. This model has four inputs plus the intercept.

```r
m2 <- lm(log(medianHouseValue) ~
           latitude +
           longitude +
           I(households/population) +
           medianIncome,
         data = train)
summary(m2)
```

```
##
## Call:
## lm(formula = log(medianHouseValue) ~ latitude + longitude + I(households/population) +
##     medianIncome, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.63222 -0.18761 -0.00298  0.18787  1.61447
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -12.110093   0.382518  -31.66   <2e-16 ***
## latitude                  -0.284438   0.003986  -71.36   <2e-16 ***
## longitude                 -0.277070   0.004290  -64.58   <2e-16 ***
## I(households/population)   1.291858   0.035490   36.40   <2e-16 ***
## medianIncome               0.187073   0.002122   88.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3174 on 9833 degrees of freedom
## Multiple R-squared:  0.6441, Adjusted R-squared:  0.644
## F-statistic:  4449 on 4 and 9833 DF,  p-value: < 2.2e-16
```

```r
plot(fitted(m2), resid(m2), cex = .1)
abline(h = 0)
```



Model `m3` is the best model with the same number of variables as `m1`. This model has eight inputs plus the intercept.

```r
m3 <- lm(log(medianHouseValue) ~
           latitude +
           longitude +
           housingMedianAge +
           I(households/totalRooms) +
           I(log(totalBedrooms/population)) +
           households +
           I(population/households) +
           medianIncome,
         data = train)
summary(m3)
```

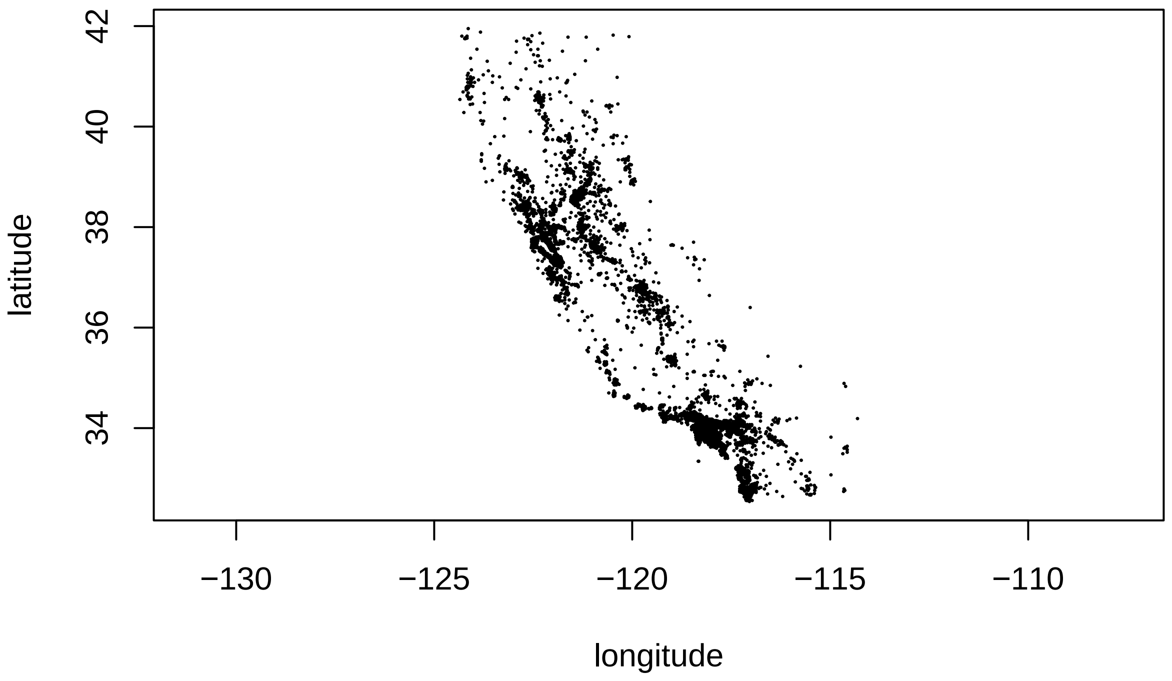
```
##
## Call:
## lm(formula = log(medianHouseValue) ~ latitude + longitude + housingMedianAge +
##     I(households/totalRooms) + I(log(totalBedrooms/population)) +
##     households + I(population/households) + medianIncome, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.98732 -0.18188  0.00065  0.17970  1.81901
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -1.003e+01  4.008e-01 -25.026   <2e-16 ***
## latitude                         -2.610e-01  4.407e-03 -59.229   <2e-16 ***
## longitude                        -2.557e-01  4.637e-03 -55.142   <2e-16 ***
## housingMedianAge                  3.714e-03  2.848e-04  13.041   <2e-16 ***
## I(households/totalRooms)          1.184e+00  7.285e-02  16.254   <2e-16 ***
## I(log(totalBedrooms/population))  4.193e-01  1.087e-02  38.585   <2e-16 ***
## households                       8.420e-05  8.856e-06   9.508   <2e-16 ***
## I(population/households)         3.990e-03  4.186e-04   9.532   <2e-16 ***
## medianIncome                     2.203e-01  2.679e-03  82.241   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3098 on 9829 degrees of freedom
## Multiple R-squared:  0.6611, Adjusted R-squared:  0.6608
## F-statistic:  2397 on 8 and 9829 DF,  p-value: < 2.2e-16
```

As the `regsubsets()` output above shows, one could consider even larger models. However, the improvement in the adjusted R^2 value is small and the models become more complicated. I will not consider larger models here.

Role of Longitude and Latitude

The variables `longitude` and `latitude` represent to centre location of each area in the census. If we plot these values, we can just make out the sketch of a map of California:

```
plot(train$longitude, train$latitude, cex = .1, asp=1/cos(38*pi/180),
     xlab = "longitude", ylab = "latitude")
```



Since house prices don't depend directly on the location of the house, but rather on the socio-economic structure of the area, it might make sense to replace the two variables `longitude` and `latitude` by a categorical variable which, which indicates geographical regions.

As an example, here I use the six largest cities in California as regions. After looking up longitude and latitude of these cities, and after some trial and error, we use the following regions (LA = Los Angeles, SD = San Diego, SJ = San Jose, SF = San Francisco, FR = Fresno, SA = Sacramento):

```
area <- rep("other", nrow(train))
area[which((train$latitude - 34.05)^2 + (train$longitude + 118.24)^2 < 0.40)] = "LA"
area[which((train$latitude - 32.72)^2 + (train$longitude + 117.16)^2 < 1.06)] = "SD"
area[which((train$latitude - 37.34)^2 + (train$longitude + 121.89)^2 < 2.40)] = "SJ"
area[which((train$latitude - 37.77)^2 + (train$longitude + 122.42)^2 < 0.16)] = "SF"
```

```r
area[which((train$latitude - 36.74)^2 + (train$longitude + 119.79)^2 < 1.55)] = "FR"
area[which((train$latitude - 38.58)^2 + (train$longitude + 121.49)^2 < 1.29)] = "SA"
area <- factor(area, levels = c("other", "LA", "SD", "SJ", "SF", "FR", "SA"))

m4 <- lm(log(medianHouseValue) ~
           I(households/population) +
           medianIncome +
           area,
         data = train)
summary(m4)
```

```
##
## Call:
## lm(formula = log(medianHouseValue) ~ I(households/population) +
##     medianIncome + area, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70278 -0.18036 -0.01767  0.17275  1.72349
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              10.600710   0.016351  648.31   <2e-16 ***
## I(households/population)  1.309560   0.034644   37.80   <2e-16 ***
## medianIncome              0.186152   0.002059   90.39   <2e-16 ***
## areaLA                    0.468049   0.009218   50.78   <2e-16 ***
## areaSD                    0.275225   0.012879   21.37   <2e-16 ***
## areaSJ                    0.463523   0.012989   35.69   <2e-16 ***
## areaSF                    0.732193   0.015979   45.82   <2e-16 ***
## areaFR                   -0.278712   0.014921  -18.68   <2e-16 ***
## areaSA                    0.111299   0.010244   10.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3063 on 9829 degrees of freedom
## Multiple R-squared:  0.6686, Adjusted R-squared:  0.6683
## F-statistic:  2479 on 8 and 9829 DF,  p-value: < 2.2e-16
```

Comparison of Two Models

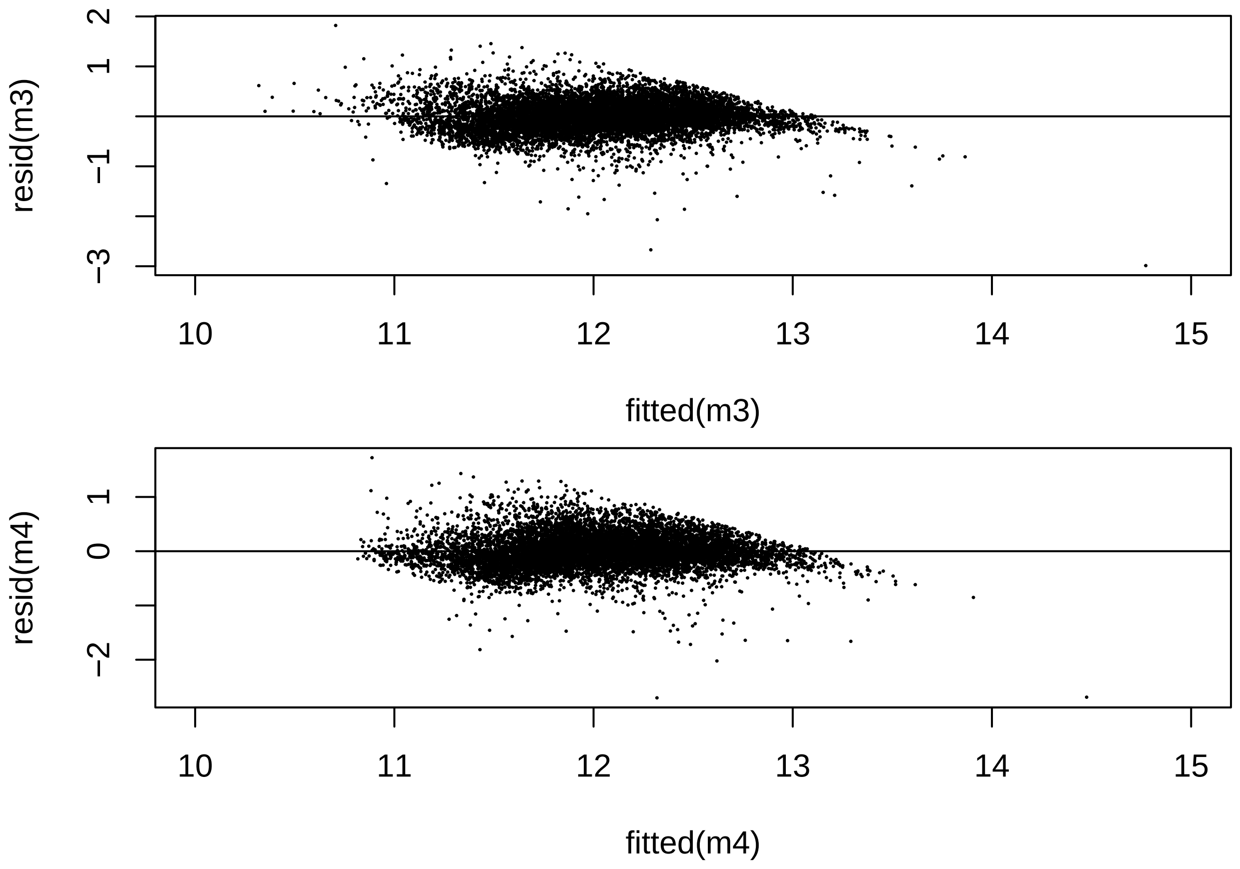
Here I will compare the models `m3` and `m4` from above:

```r
m3 <- lm(log(medianHouseValue) ~
           latitude +
           longitude +
           housingMedianAge +
           I(households/totalRooms) +
           I(log(totalBedrooms/population)) +
           households +
           I(population/households) +
           medianIncome,
         data = train)
```

```r
m4 <- lm(log(medianHouseValue) ~
           I(households/population) +
           medianIncome +
           area,
         data = train)
```

We have already seen that the adjusted R^2 value for both models are similar, with `m4` being slightly better. Here we also compare the residual plots:

```
par(mfrow = c(2,1), mai = c(0.8, 0.8, 0.1, 0.1))
plot(fitted(m3), resid(m3), cex = .1, xlim=c(10, 15))
abline(h = 0)
plot(fitted(m4), resid(m4), cex = .1, xlim=c(10, 15))
abline(h = 0)
```



Both plots look acceptable. Maybe `m4` is slightly better for the lower range of fitted values?

Between these two models, I would prefer `m4` because it is simpler and seems to have slightly better model fit.

	`m3`	`m4`
number of variables	8	3
columns in the design matrix	9	9
adjusted R^2	0.6608	0.6683
residual plot	acceptable	acceptable

Interpretation of `m4`

Model `m4` as fitted is

$$\log(\text{median house value}) \approx 10.60 + c_{\text{area}} + 1.30\,\frac{\text{households}}{\text{population}} + 0.18\,\text{median income}$$

where c_{area} is given in the following table:

area	LA	SD	SJ	SF	FR	SA	other
c_{area}	0.47	0.28	0.46	0.73	-0.28	0.11	0

Exponentiating then gives the model for median house values in USD:

$$\text{median house value} \approx 40{,}163.35\,\exp(c_{\text{area}})\,\exp\left(1.30\,\frac{\text{households}}{\text{population}} + 0.18\,\text{median income}\right)$$

The individual terms in this model are easy to interpret:

- $\exp(c_{\text{area}})$ accounts for the effect of a house being in one of the large cities. For example, in the model, houses in Los Angeles are about $\exp(0.47) = 1.60$ times more expensive than in other parts of California.
- The term households/population is typically less than one and measures how many people live in one household. Low values indicate that many people share one household. Higher values of the ratio indicate the opposite and may be a proxy for city living or well-off areas, both of which may be correlated with higher house prices. Thus it makes sense that the coefficient in front of this term is positive.
- We have already seen in task 1 that house prices increase with income in the area, and thus it is no surprise the coefficient in front of median income is also positive.

Marking Criteria

- A reasonable effort has been made at finding good variables for the model. (It is not required that all details of the search for a good model are included in the report.)
- A reasonable model has been fitted. I would expect the fitted model to be better than the base model `m0`, described above.
- Some connection between Longitude/Latitude and the geography of California has been discussed. (It is not required for this to be used in the model.)
- There is a meaningful comparison of at least two models.
- One preferred model is chosen and this model is clearly described (not only in the form of R output).
- Some connection is made between the fitted regression coefficients and the real world.

Task 3 (7 marks)

Now we load the test data set into R:

```r
test <- read.csv("https://www1.maths.leeds.ac.uk/~voss/data/housing/test.csv")
```

Using the model `m4` fitted above, we predict the outputs for these new data. Before we can do this, we need to compute the `area` values for the test data set:

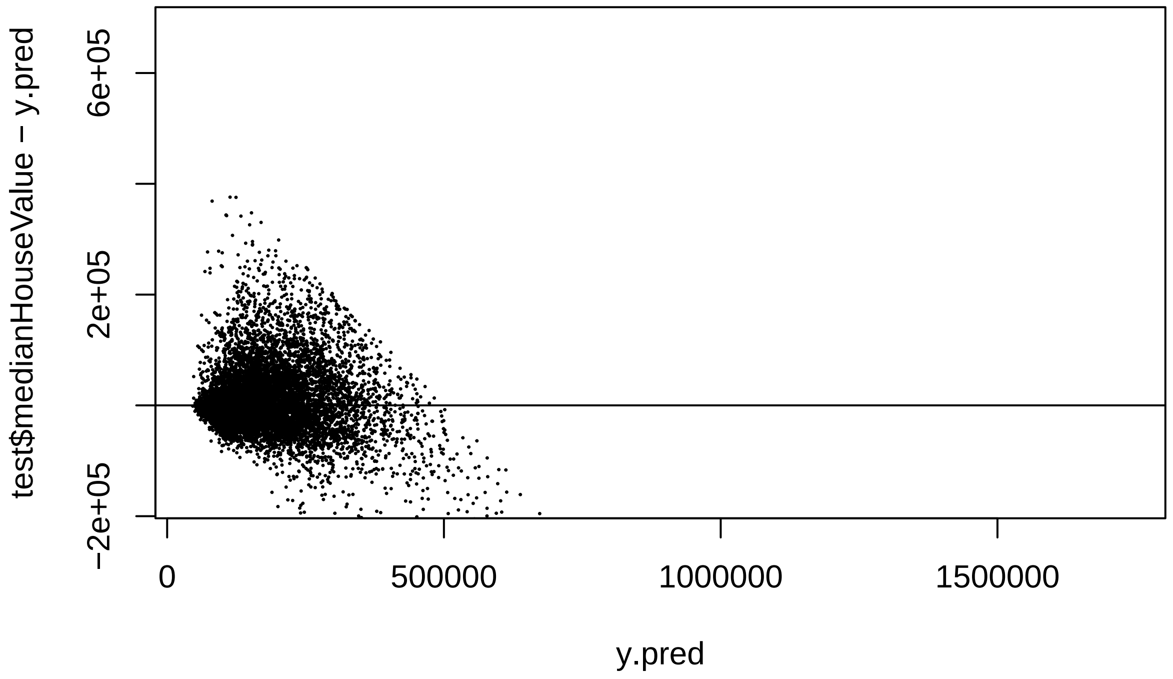
```r
area <- rep("other", nrow(test))
area[which((test$latitude - 34.05)^2 + (test$longitude + 118.24)^2 < 0.40)] = "LA"
area[which((test$latitude - 32.72)^2 + (test$longitude + 117.16)^2 < 1.06)] = "SD"
area[which((test$latitude - 37.34)^2 + (test$longitude + 121.89)^2 < 2.40)] = "SJ"
area[which((test$latitude - 37.77)^2 + (test$longitude + 122.42)^2 < 0.16)] = "SF"
area[which((test$latitude - 36.74)^2 + (test$longitude + 119.79)^2 < 1.55)] = "FR"
area[which((test$latitude - 38.58)^2 + (test$longitude + 121.49)^2 < 1.29)] = "SA"
area <- factor(area, levels = c("other", "LA", "SD", "SJ", "SF", "FR", "SA"))
```

Now we can apply the model to the new data, and transform back the result to get median house prices (instead of logarithms thereof):

```r
log.y.pred <- predict(m4, newdata = test)
y.pred <- exp(log.y.pred)
```

As a sanity check, we consider a residual plot for the test data set:

```r
plot(y.pred, test$medianHouseValue - y.pred, asp = 1, cex = 0.1,
     ylim = range(test$medianHouseValue))
abline(h = 0)
```



The plot looks reasonable enough to make me think that I have applied the model correctly, but there is a lot of noise in the model. Also, some of the predicted values are extremely high (due to the exponentiation).

Finally, we compute the required Mean Squared Error:

```r
MSE <- mean((y.pred - test$medianHouseValue)^2)
cat("MSE =", MSE, "\n")
```

```
## MSE = 4545006070
```

```r
cat("RMSE =", sqrt(MSE), "\n")
```

```
## RMSE = 67416.66
```

As a squared quantity, the MSE is of course very large. To make interpretation easier we also compute the

RMSE. At 67,416 dollar, the RMSE still seems high, but this might be a consequence of the "outlier" in the predicted values. To verify this hypothesis, we recalculate the RMSE using the median instead of the mean:

```r
MedianSE <- median((y.pred - test$medianHouseValue)^2)
cat("RMedianSE =", sqrt(MedianSE), "\n")
```

```
## RMedianSE = 29319.3
```

The value is now much lower, which confirms that the MSE has been strongly affected by the outlier.

Marking Criteria

- The test data set is loaded correctly.
- The pre-fitted model is applied correctly. The model is **not** fitted to the test data set.
- If needed, the predicted values are transformed back into median house prices, undoing any transformation applied to the outputs when fitting the model.
- The MSE is computed correctly.
- There is some meaningful discussion of the resulting MSE value.

Task 4 (8 marks, level 5 only)

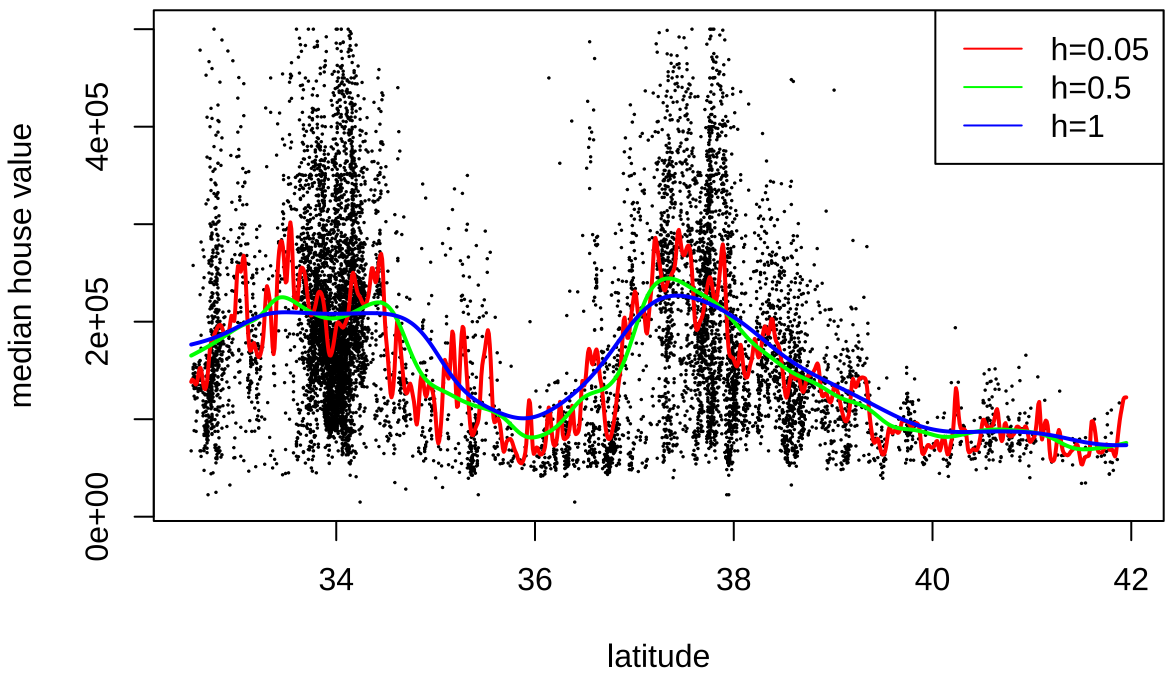
The Nadaraya-Watson estimator is discussed in the level 5 lecture notes. To compute the estimator we need to choose a kernel and the bandwidth. For simplicity, we choose a Gaussian kernel here.

To choose the bandwidth, we start with a simple experiment, plotting the data with different bandwidths:

```r
x <- train$latitude
y <- train$medianHouseValue
plot(x, y, cex=.1, xlab = "latitude", ylab = "median house value")

lines(ksmooth(x, y, kernel = "normal", bandwidth = 0.05, n.points = 1000),
      col = "red", lwd = 2)
lines(ksmooth(x, y, kernel = "normal", bandwidth = 0.5, n.points = 1000),
      col = "green", lwd = 2)
lines(ksmooth(x, y, kernel = "normal", bandwidth = 1, n.points = 1000),
      col = "blue", lwd = 2)

legend("topright", legend = c("h=0.05", "h=0.5", "h=1"),
       col = c("red", "green", "blue"), lty = 1)
```



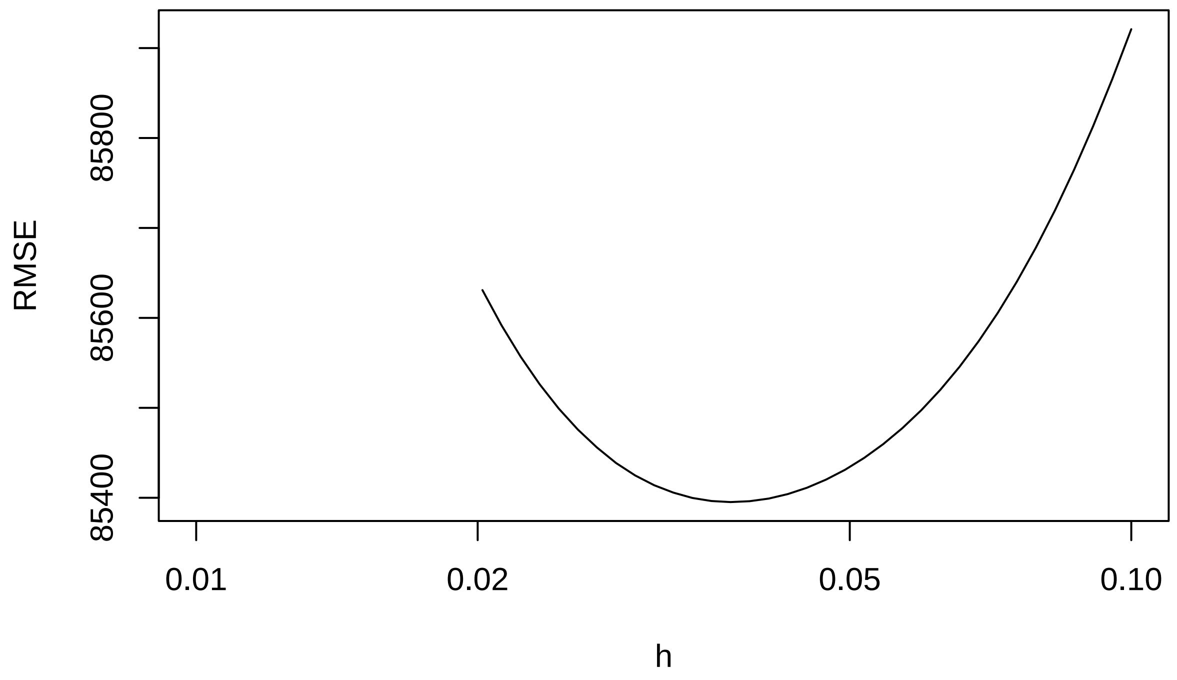
There are different possibilities for chosing the bandwidth h:

- The most proper way would be to use cross-validation to choose h, by minimising the cross-validated MSE. This is discussed in section 8.1 of the lecture notes.
- We could use the test-set to choose h, by minimising the test-set MSE.
- We could choose h visually using plots like the one above.
- In section 4.3 of the notes we have seen a heuristic formula for choosing h for kernel density estimation. It is possible that the resulting h is also a good choice for the Nadaraya-Watson estimator.

Here I try the second method, using the test set to choose h:

```r
# ksmooth wants the x-values to be sorted in increasing order
idx <- order(test$latitude)
x.test <- test$latitude[idx]
y.test <- test$medianHouseValue[idx]

k <- 50
h <- exp(seq(log(0.01), log(0.1), length.out = k))
RMSE <- numeric(k)
for (i in 1:k) {
  m <- ksmooth(x, y, kernel = "normal", bandwidth = h[i], x.points = x.test)
  RMSE[i] <- sqrt(mean((m$y - y.test)^2))
}
plot(h, RMSE, type="l", log = "x")
```



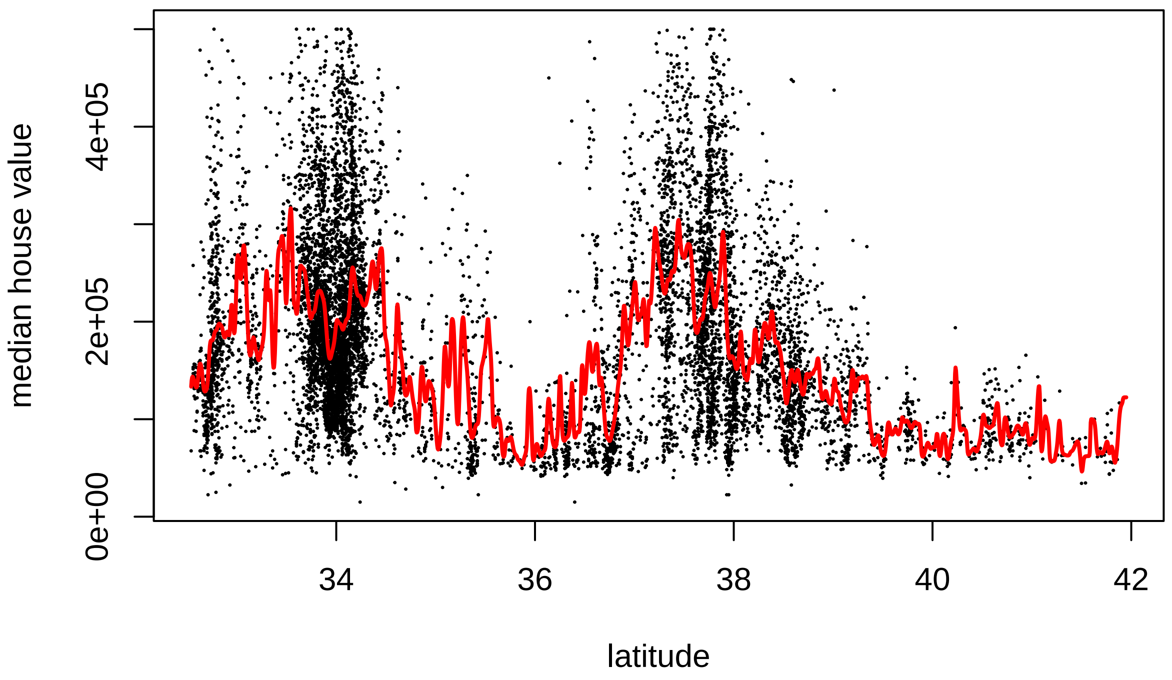
The best h obtained this way is $h = 0.037$:

```
h.opt <- h[which.min(RMSE)]
h.opt
```

```
## [1] 0.03727594
```

The corresponding Nadaraya-Watson estimator is:

```
x <- train$latitude
y <- train$medianHouseValue
plot(x, y, cex=.1, xlab = "latitude", ylab = "median house value")
lines(ksmooth(x, y, kernel = "normal", bandwidth = h.opt, n.points = 2000),
      col = "red", lwd = 2)
```



The resulting line is quite "wiggly", and less smooth than what I would have chosen manually based on a plot. Maybe the peaks in this plot represent the latitude of towns and cities in California?

Marking Criteria:

- The NW estimator is computed correctly (e.g. using the `ksmooth()` function).
- The bandwidth is chosen carefully, and the choice is justified.
- There is some discussion of the resulting NW estimator.