Hands-on Machine Learning with R

Linear Regression

Linear regression is one of the simplest algs for supervised learning. But it's a good starting point and many more complex methods can be seen as extensions of it.

Prerequisites

Adding `vip` packages for interpretability of variable importance. Ames data set from before.

Simple Linear Regresison

SLR assumes the relationship between two continuous variables is at least approximately linear.

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \qquad for \quad i = 1, 2, ..., n,$$

Where Y_i represents the response/target variable, X_i is the i^{th} feature value and teh betas are fixed but unknown constants (coefficients or parameters), representing the intercept and the slope.

The ϵ_i term represents noise or random error. Here we assume the errors have a mean of zero and constant variance σ^2. This is denoted as $\overset{iid}{\sim} N(0, \sigma^2)$. Since the errors are centered on zero - the expected value $E(\epsilon_i) = 0$, linear regression is really a problem of estimating the conditional mean:

$$E(Y_i|X_i) = \beta_0 + \beta_1 X_i$$

WHich we can shorten to just $E(Y)$. So the interpretation is in therms of *average responses*. E.g. β_0 is the average response value when $X = 0$ - sometimes referred to as the *bias term* and β_1 is the increase in the average response if X increases by one unit, aka the *rate of change*.

Estimation

We want the best fitting line, but what is the best fit? The most common way, called *Oridnary least squares* (OLS) is to minimise the *residual sum of squares*:

$$RSS(\beta_0, \beta_1) = \sum_{i=1}^{n} [Y_i - (\beta_0 + \beta_1 X_i)]^2 = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2.$$

We denote the OLS stimates of the coefficients as $\hat{\beta}_0$ and $\hat{\beta}_1$. Once we have the estimated regression equation, we can predict values of Y for X_{new}:

$$\hat{Y}_{new} = \hat{\beta}_0 + \hat{\beta}_1 X_{new}$$

Where $\hat{Y}_{new}$ is the estimated mean response at $X = X_{new}$.

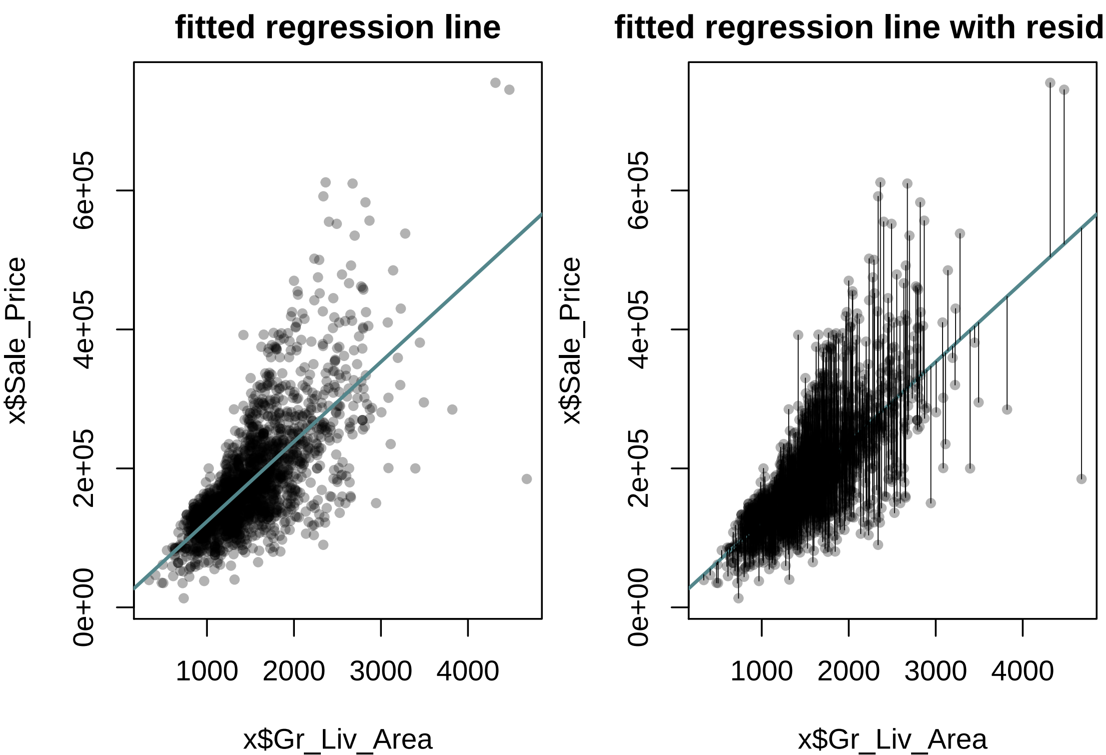
So let's try modelling the ames data relationship between the sale price and the above ground living area. This link)has good info on visualising residuals.

```
model1 <- lm(Sale_Price ~Gr_Liv_Area, data = ames_train)

# let's have a look at the resisiduals and plot them

x <- ames_train
x$predicted <- predict(model1)   # Save the predicted values
```

```r
x$residuals <- residuals(model1)

# this is what the data looks like
x %>% select(Sale_Price, predicted, residuals) %>% head()
```

```
## # A tibble: 6 x 3
##   Sale_Price predicted residuals
##        <int>     <dbl>     <dbl>
## 1     215000   198968.    16032.
## 2     105000   111662.    -6662.
## 3     172000   161403.    10597.
## 4     195500   192994.     2506.
## 5     213500   162437.    51063.
## 6     236500   194373.    42127.
```

```r
# here are some plots
par(mfrow = c(1,2))
par(mar = c(4,4,2,0.1)+0.10)
plot(x$Gr_Liv_Area, x$Sale_Price, col = alpha("black", 0.3), pch = 20,
     main = "fitted regression line")
abline(model1, col = "cadetblue4", lwd = 2)
plot(x$Gr_Liv_Area, x$Sale_Price, col = alpha("black", 0.3), pch = 20,
     main = "fitted regression line with residuals")
abline(model1, col = "cadetblue4", lwd = 2)
for (i in 1: nrow(x)){
lines(c(x$Gr_Liv_Area[i], x$Gr_Liv_Area[i]),
      c(x$Sale_Price[i], x$predicted[i]),
      lwd = 0.5)}
```



Use `coef()` and `summary()` to have a look a the coefficients. !!! I don't get the same data, even though i have the same seed in the split?

```r
coef(model1)
```

```
## (Intercept) Gr_Liv_Area
##    8732.938     114.876
```

```r
summary(model1)
```

```
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -361143  -30668   -2449   22838  331357
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8732.938   3996.613   2.185    0.029 *
## Gr_Liv_Area  114.876      2.531  45.385   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 56700 on 2051 degrees of freedom
## Multiple R-squared:  0.5011, Adjusted R-squared:  0.5008
## F-statistic:  2060 on 1 and 2051 DF,  p-value: < 2.2e-16
```

```r
# glimpse(model1)
```

So we estimate that an increase in area by one square foot increases the selling price by 114.88$. SO nice and intuitive.

One drawback of using least squares is that we only have estimates of the coefficients, but not of the error variance σ^2. LS makes no assumptions about the random errors, so we cannot estimate σ^2.

An alternative is to use *maximum likelihood* estimation (ML) to estimate σ^2 – which we need to characterise the variability of our model. For ML we have to assume a particular distribution of the errors, most commonly that they are normally distributed. Under these assumptions the estimate of the error variance is

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \frac{1}{n-p}\sum_{i=1}^{n} r_i^2$$

Where r_i is the residual of the i^{th} observation. and p is the number of parameters or coefficients in the model. $\hat{\sigma}^2$ is also known as the mean squared error (MSE) and it's square root is the RMSE, and you can get it out of an `lm` object using `sigma()`

```r
sigma(model1)
```

```
## [1] 56704.78
```

```r
sigma(model1)^2
```

```
## [1] 3215432370
```

!!! the sigma is slightly different from the RMSE reported in the summary, not sure why, same in book.

Inference

The coefficients are only point estimates, so that's not super useful without a measure of variability. This is usually measured with a *standard error* (SE), the square root of it's variance.

If we assume the errors are distributed $\overset{iid}{\sim} N(0, \sigma^2)$,then the SEs for the coefficients are simple and are expressied under the `Std. Error` heading in the summary for the model.

From the SE we can also do a t-test to see if the coefficients are statistically significantly different from zero. (!!! stantistically significant from zero is probably wrong).

The t-statistic is simply the estimated coefficient divided by the SE, which measures the number of standard deviations each coefficient is away from zero. The p-values are reported in the same table.

Under these same assumptions we can also derive the confidence intervals for the β coefficients. The formual is:

$$\beta_j \pm t_{1-\alpha/2,n-p} \hat{SE}(\hat{\beta}_j)$$

In R you can construct them using `confitn()`

```
confint(model1, level = 0.95)
```

```
##                 2.5 %     97.5 %
## (Intercept) 895.0961 16570.7805
## Gr_Liv_Area 109.9121   119.8399
```

```
# or if you wanna spell it out:
coef(model1)[2] - qt(0.975, model1$df.residual)*coef(summary(model1))[2,2]
```

```
## Gr_Liv_Area
##    109.9121
```

```
coef(model1)[2] + qt(0.975, model1$df.residual)*coef(summary(model1))[2,2]
```

```
## Gr_Liv_Area
##    119.8399
```

So with 95% confidence we estimate that the mean sale price goes up between 109 and 119$ for each additional square foot.

Don't forget that these SEs and t-stats etc in the summary are based on the following assumptions:

1. Independent observations
2. The random errors have mean zero, and constant variance
3. The random errors are normally distributed

If your data deviate from these assumptions, there are some remedial actions you can take..

Multiple linear regression

Extend the simple linear regression with more predictors to see e.g. how are and year built are (linearly) related to the sales price using *mulitple linear regression* (MLR).

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_2 + \epsilon_i, \qquad for \quad i = 1, 2, ..., n,$$

Which you do in R by using `+` to separate predictors:

```
(model2 <- lm(Sale_Price ~ Gr_Liv_Area + Year_Built, data = ames_train))
```

```
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area + Year_Built, data = ames_train)
##
## Coefficients:
## (Intercept)  Gr_Liv_Area   Year_Built
##  -2.123e+06    9.918e+01    1.093e+03
```

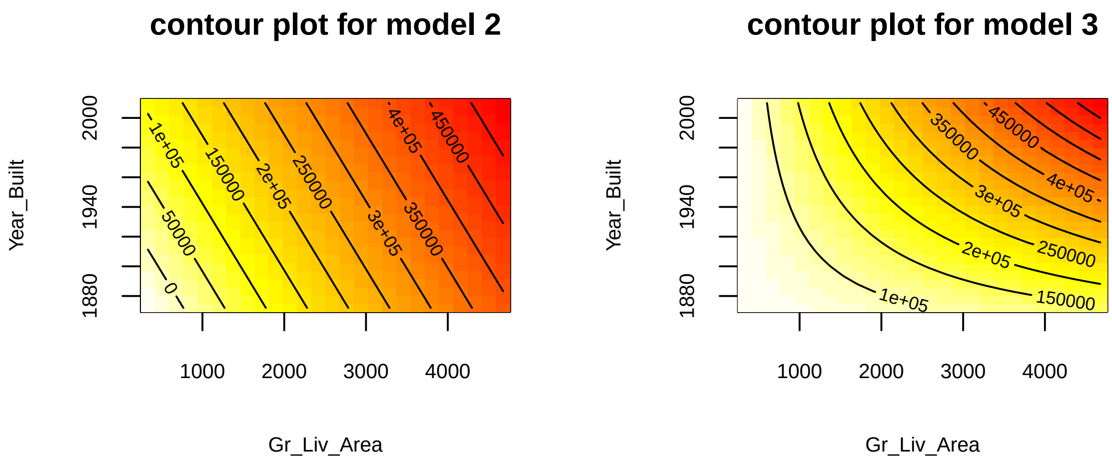


Figure 1: Contour plot of the fitted regression surface

```r
# or use update
(model2 <- update(model1, .~. + Year_Built))
```

```
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area + Year_Built, data = ames_train)
##
## Coefficients:
## (Intercept)  Gr_Liv_Area   Year_Built
##  -2.123e+06    9.918e+01    1.093e+03
```

So holding the year constant, each additional square foot of living area increses the mean selling price by 99$. And holding the area constant, each additional year the home is newer by increases the mean price by 1093$. Here are some contour plots:

The left one only has main effects and is therefore flat. Including interaction effects models curvature: the effect of one predictor now depend on the level of the other. So in our example this would mean including the product of both predictors:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_2 + \beta_3 X_1 X_2 \epsilon_i, \qquad for \quad i = 1, 2, ..., n,$$

In R the formula is either `y ~ x1 + x2 ~ x1:x2` or `y ~ x1 * x2`.

Note the *hierarchy principle* which means that any lower order terms corresponding to the interaction term must also be included in the model.

You can include as many predictors as you like - as long as you have more observations than predictors! (So in wide tables you cannot include all of them!). These can also be interactions, or transoformations: e.g. $X_3 = X_1 X2$ or $X_4 = \sqrt(X_3)$. Of course after two dimensions visualisation becomes impractical, because we have a hyperplane of best fit.

We can try all of the predictors in the data set and clean up the output using the `broom` package: (!!! again, the results are even more different than before).

```r
model4 <- lm(Sale_Price ~ ., data = ames_train)

broom::tidy(model4)
```

```
## # A tibble: 283 x 5
##    term                                      estimate std.error statistic p.value
##    <chr>                                        <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)                                -5.61e6 11261881.    -0.498   0.618
```

```
##  2 MS_SubClassOne_Story_1945_and_Older   3.56e3     3843.     0.926   0.355
##  3 MS_SubClassOne_Story_with_Finished~   1.28e4    12834.     0.997   0.319
##  4 MS_SubClassOne_and_Half_Story_Unfi~   8.73e3    12871.     0.678   0.498
##  5 MS_SubClassOne_and_Half_Story_Fini~   4.11e3     6226.     0.660   0.509
##  6 MS_SubClassTwo_Story_1946_and_Newer  -1.09e3     5790.    -0.189   0.850
##  7 MS_SubClassTwo_Story_1945_and_Older   7.14e3     6349.     1.12    0.261
##  8 MS_SubClassTwo_and_Half_Story_All_~  -1.39e4    11003.    -1.27    0.206
##  9 MS_SubClassSplit_or_Multilevel       -1.15e4    10512.    -1.09    0.276
## 10 MS_SubClassSplit_Foyer               -4.39e3     8057.    -0.545   0.586
## # ... with 273 more rows
```

Assessing model accuracy

So now we have three main effects models, a single predictor one, one with two predictors and one with all of the features. WHich is best? Let's use RMSE and cross-validation. (this means resampling from the training dataset and validating on sub-folds, and then taking the average RMSE, instead of just the RMSE of the models as given in the summary()).

So we can use `caret::train()` to train the model using cross-validation, which is not available directly in the `lm()` funciton

```
# Train model using 10-fold cross-validation
set.seed(123)  # for reproducibility
(cv_model1 <- train(
  form = Sale_Price ~ Gr_Liv_Area,
  data = ames_train,
  method = "lm",
  trControl = trainControl(method = "cv", number = 10)
))
```

```
## Linear Regression
##
## 2053 samples
##    1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1846, 1848, 1848, 1848, 1848, 1848, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   56410.89  0.5069425  39169.09
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

So when applied to unseen data, model1 is on average $56,600 off the mark. Let's perform cv on the other two models as well.

```
# Train model using 10-fold cross-validation
set.seed(123)  # for reproducibility
(cv_model2 <- train(
  form = Sale_Price ~ Gr_Liv_Area + Year_Built,
  data = ames_train,
  method = "lm",
  trControl = trainControl(method = "cv", number = 10)
))
```

```
## Linear Regression
##
```

```
## 2053 samples
##    2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1846, 1848, 1848, 1848, 1848, 1848, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   46292.38  0.6703298  32246.86
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
set.seed(123)  # for reproducibility
(cv_model3 <- train(
  form = Sale_Price ~ .,
  data = ames_train,
  method = "lm",
  trControl = trainControl(method = "cv", number = 10)
))
```

```
## Linear Regression
##
## 2053 samples
##   80 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1846, 1848, 1848, 1848, 1848, 1848, ...
## Resampling results:
##
##   RMSE   Rsquared   MAE
##   26098  0.8949642  16258.84
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# collect results from all three resamplings
summary(resamples(list(
  model1 = cv_model1,
  model2 = cv_model2,
  model3 = cv_model3
)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(model1 = cv_model1, model2
##  = cv_model2, model3 = cv_model3)))
##
## Models: model1, model2, model3
## Number of resamples: 10
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## model1 34457.58 36323.74 38943.81 39169.09 41660.81 45005.17    0
## model2 28094.79 30594.47 31959.30 32246.86 34210.70 37441.82    0
## model3 12458.27 15420.10 16484.77 16258.84 17262.39 19029.29    0
##
## RMSE
```

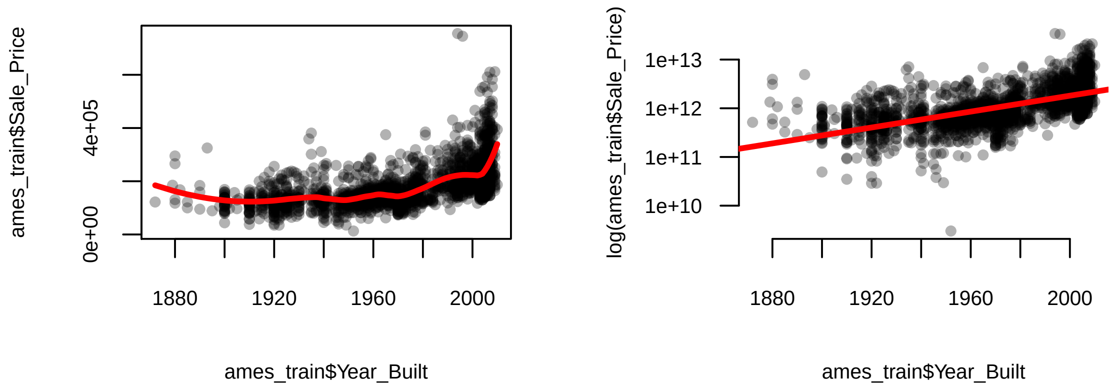


Figure 2: Transforming the target var to make the relationship more linear

```
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## model1 47211.34 52363.41 54948.96 56410.89 60672.31 67679.05    0
## model2 37698.17 42607.11 45407.14 46292.38 49668.59 54692.06    0
## model3 20844.33 22581.04 24947.45 26098.00 27695.65 39521.49    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## model1 0.3598237 0.4550791 0.5289068 0.5069425 0.5619841 0.5965793    0
## model2 0.5714665 0.6392504 0.6800818 0.6703298 0.7067458 0.7348562    0
## model3 0.7869022 0.9018567 0.9104351 0.8949642 0.9166564 0.9303504    0
```

The function `caret::resamples()` allows you to compare the results of the resamplings. (!!! Again, my results are quite different from the book) The two predictor model has an average out of sample RMSE 46,292, and the all predictor model it's 26,098. Judging only by RMSE, modle 3 is the best.

Model concerns

There are several strong assumptions required by linear regression, that are often violated. What are they and what can you do about them?

1. **linearity of relationship**: if the relationship isn't linear, there are still transformations that could make it so. See e.g. the relatioship between the year the house was built and the price. It's not linear, but log-transforming the target variable can make it more so.