Machine Learning: Regression in R

Jan-Philipp Kolb

16 Januar, 2020

Why a part on linear regression

- OLS can be seen as a simple machine learning technique
- Some other machine learning concepts are based on regression (e.g. regularization).
- We would like to remind you how simple regression works in R.
- We also want to show the constraints
- In a next step we will learn, how to coop with these constraints

The Ames Iowa Housing Data

The dataset describes the sale of individual residential property in Ames, Iowa from 2006 to 2010.

```
ames_data <- AmesHousing::make_ames()
```

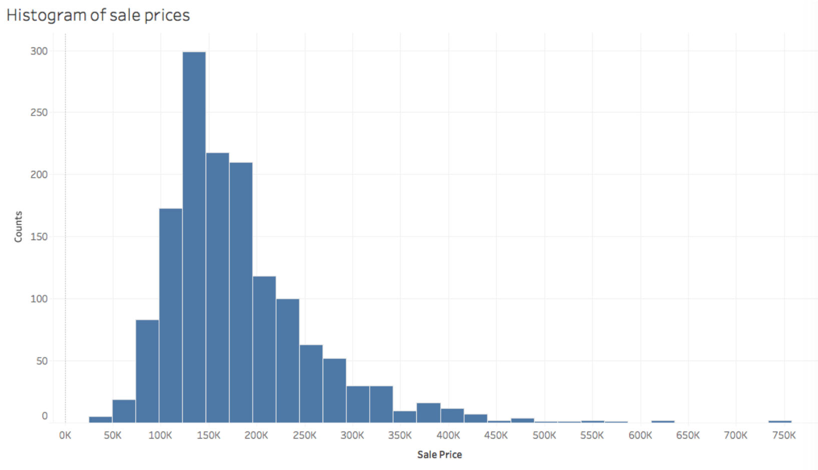
Some Variables

- `Gr_Liv_Area`: Above grade (ground) living area square feet
- `TotRms_AbvGrd`: Total rooms above grade (does not include bathrooms
- `MS_SubClass`: Identifies the type of dwelling involved in the sale.
- `MS_Zoning`: Identifies the general zoning classification of the sale.
- `Lot_Frontage`: Linear feet of street connected to property
- `Lot_Area`: Lot size in square feet
- `Street`: Type of road access to property
- `Alley`: Type of alley access to property
- `Lot_Shape`: General shape of property
- `Land_Contour`: Flatness of the propert

Exercise: regression Ames housing data

1) Install the package `AmesHousing` and create a **processed version** of the Ames housing data with (at least) the variables `Sale_Price`, `Gr_Liv_Area` and `TotRms_AbvGrd`
2) Create a regression model with `Sale_Price` as dependent and `Gr_Liv_Area` and `TotRms_AbvGrd` as independent variables. Then create seperated models for the two independent variables. Compare the results. What do you think?

The sale price



A simple regression model

Dependent variable - Sale_Price

- the sale price of houses

Independent variable - Gr_Liv_Area

```
m1 <- lm(Sale_Price ~ Gr_Liv_Area,data=ames_data)
m1
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_data)
##
## Coefficients:
## (Intercept)  Gr_Liv_Area
##     13289.6        111.7
```

Get the model summary

```
summary(m1)
```

```
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -483467  -30219   -1966   22728  334323
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13289.634   3269.703   4.064 4.94e-05 ***
## Gr_Liv_Area   111.694      2.066  54.061  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
##
## Residual standard error: 56520 on 2928 degrees of freedom
```

The model formula

Model without intercept

```
m2 <- lm(Sale_Price ~ - 1 +Gr_Liv_Area,data=ames_data)
summary(m2)$coefficients
##             Estimate Std. Error  t value Pr(>|t|)
## Gr_Liv_Area 119.6517  0.6615846 180.8563        0
```

Adding further variables

```
m3 <- lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd,
         data=ames_data)
summary(m3)$coefficients
##                  Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept)    42767.6361 4372.532783  9.780976  2.967720e-22
## Gr_Liv_Area      139.4075    3.447581 40.436332 2.058869e-284
## TotRms_AbvGrd -11025.8696 1107.960753 -9.951498  5.730058e-23
```

Further possibilities to specify the formula

Take all available predictors

```
m3_a<-lm(Sale_Price~.,data=ames_data)
```

Interaction effect

```
# effect of cyl and interaction effect:
m3a<-lm(Sale_Price~Lot_Area*Bedroom_AbvGr,data=ames_data)

# only interaction effect:
m3b<-lm(Sale_Price~Lot_Area:Bedroom_AbvGr,data=ames_data)
```

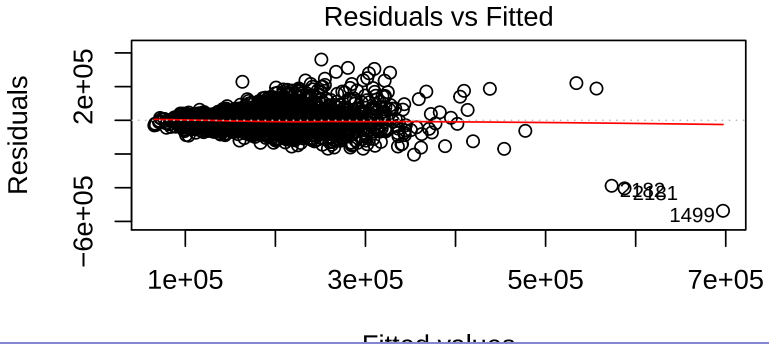
Take the logarithm

```
m3d<-lm(Sale_Price~log(Lot_Area),data=ames_data)
```

Residual plot - model assumptions violated?

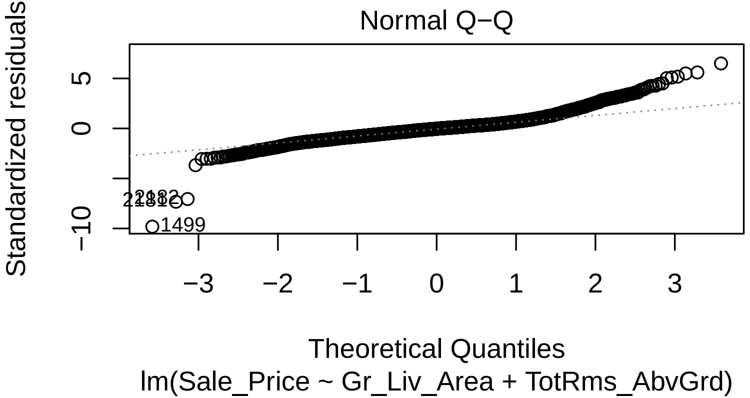
- We have model assumptions violated if points deviate with a pattern from the line

```
plot(m3,1)
```



Residual plot

```
plot(m3,2)
```



Another example for object orientation

- `m3` is now a special regression object
- Various functions can be applied to this object

```
predict(m3) # Prediction
resid(m3) # Residuals
```

```
##        1        2        3        4        5        6
## 196445.4 112547.4 161885.0 248710.6 203707.3 189196.2
```

```
##          1          2          3          4          5
##  18554.583  -7547.434  10114.975  -4710.566 -13807.284   6303
```

Make model prediction

```
pre <- predict(m1)
head(mtcars$mpg)
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

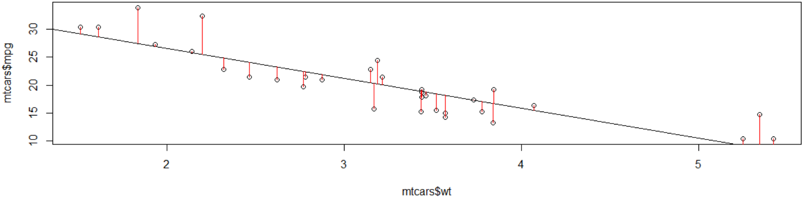
```
head(pre)
```

```
##        1        2        3        4        5        6
## 198254.9 113367.5 161731.0 248964.0 195239.2 192446.8
```

Regression diagnostic with base-R

Visualizing residuals

```
plot(mtcars$wt,mtcars$mpg)
abline(m1)
segments(mtcars$wt, mtcars$mpg, mtcars$wt, pre, col="red")
```



The mean squared error (MSE)

- The **MSE** measures the average of the squares of the errors
- **The lower the better**

```
(mse5 <- mean((mtcars$mpg -  pre)^2)) # model 5
```

```
## [1] 35866849640
```

```
(mse3 <- mean((mtcars$mpg -  predict(m3))^2))
```

```
## [1] 35971337573
```

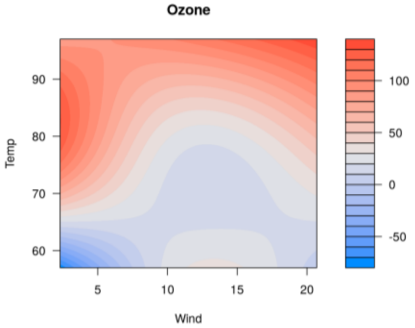
Package Metrics to compute MSE

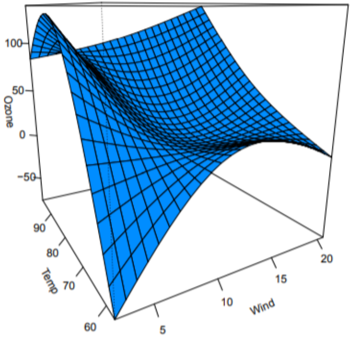
```
library(Metrics)
mse(mtcars$mpg,predict(m3))
## [1] 35971337573
```

THE VISREG-PACKAGE

```
install.packages("visreg")
```

```
library(visreg)
```

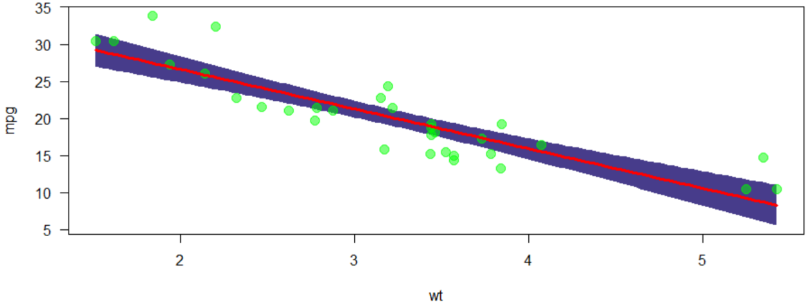




The visreg-package

- The default-argument for `type` is `conditional`.
- Scatterplot of `mpg` and `wt` plus regression line and confidence bands

```
visreg(m1, "wt", type = "conditional")
```



Regression with factors

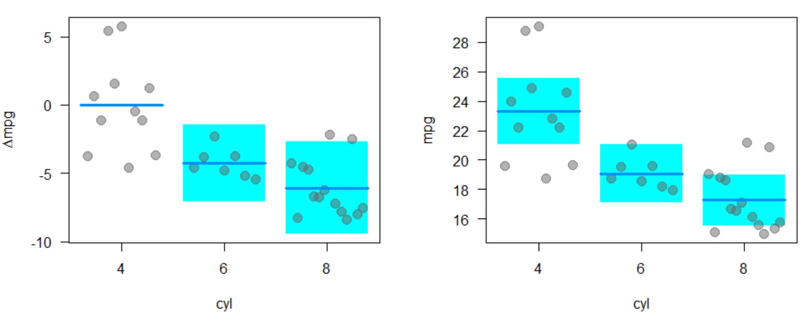
- The effects of factors can also be visualized with `visreg`:

```
mtcars$cyl <- as.factor(mtcars$cyl)
m4 <- lm(mpg ~ cyl + wt, data = mtcars)
# summary(m4)
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 33.990794  1.8877934 18.005569 6.257246e-17
## cyl6        -4.255582  1.3860728 -3.070244 4.717834e-03
## cyl8        -6.070860  1.6522878 -3.674214 9.991893e-04
## wt          -3.205613  0.7538957 -4.252065 2.130435e-04
```

Effects of factors

```
par(mfrow=c(1,2))
visreg(m4, "cyl", type = "contrast")
visreg(m4, "cyl", type = "conditional")
```



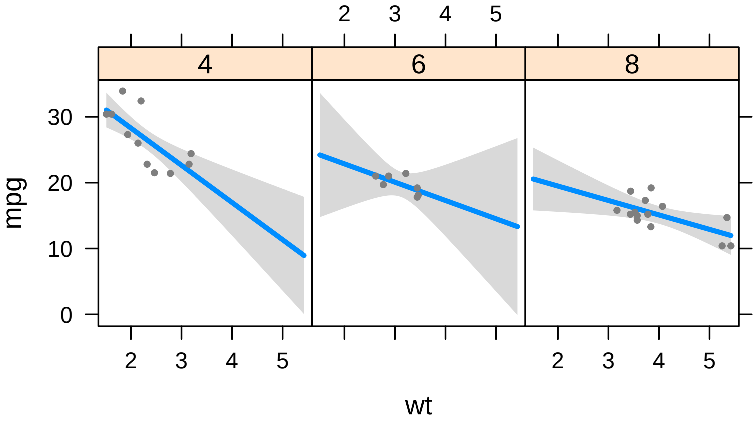
The package visreg - Interactions

```
m5 <- lm(mpg ~ cyl*wt, data = mtcars)
# summary(m5)
```

```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  39.571196   3.193940 12.3894599 2.058359e-12
## cyl6        -11.162351   9.355346 -1.1931522 2.435843e-01
## cyl8        -15.703167   4.839464 -3.2448150 3.223216e-03
## wt           -5.647025   1.359498 -4.1537586 3.127578e-04
## cyl6:wt       2.866919   3.117330  0.9196716 3.661987e-01
## cyl8:wt       3.454587   1.627261  2.1229458 4.344037e-02
```

Control of the graphic output with `layout`.

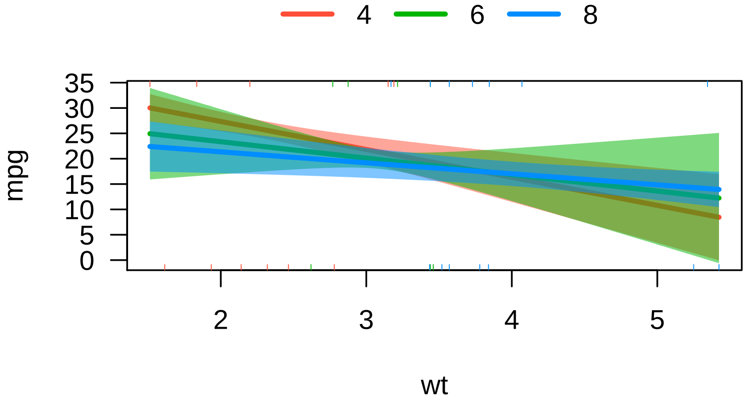
```
visreg(m5, "wt", by = "cyl",layout=c(3,1))
```



The package visreg - Interactions overlay

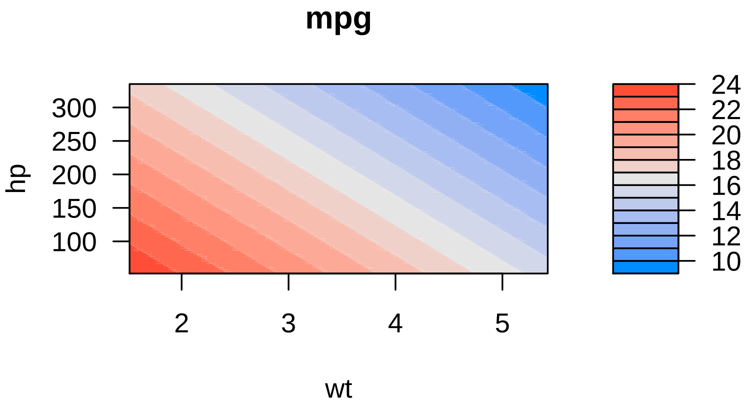
```
m6 <- lm(mpg ~ hp + wt * cyl, data = mtcars)

visreg(m6, "wt", by="cyl", overlay=TRUE, partial=FALSE)
```



The package visreg - visreg2d

```
visreg2d(m6, "wt", "hp", plot.type = "image")
```



Multicollinearity

- As p increases we are more likely to capture multiple features that have some multicollinearity.
- When multicollinearity exists, we often see high variability in our coefficient terms.
- E.g. we have a correlation of 0.801 between `Gr_Liv_Area` and `TotRms_AbvGrd`
- Both variables are strongly correlated to the response variable (`Sale_Price`).

```
ames_data <- AmesHousing::make_ames()
cor(ames_data[,c("Sale_Price","Gr_Liv_Area","TotRms_AbvGrd")])
```

```
##               Sale_Price Gr_Liv_Area TotRms_AbvGrd
## Sale_Price     1.0000000   0.7067799     0.4954744
## Gr_Liv_Area    0.7067799   1.0000000     0.8077721
## TotRms_AbvGrd  0.4954744   0.8077721     1.0000000
```

Effects of multicollinearity

```
lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data = ames_data)
```

```
##
## Call:
## lm(formula = Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data =
##
## Coefficients:
##   (Intercept)    Gr_Liv_Area  TotRms_AbvGrd
##       42767.6          139.4       -11025.9
```

- When we fit a model with both these variables we get a positive coefficient for `Gr_Liv_Area` but a negative coefficient for `TotRms_AbvGrd`, suggesting one has a positive impact to Sale_Price and the other a negative impact.

Seperated models

- If we refit the model with each variable independently, they both show a positive impact.
- The `Gr_Liv_Area` effect is now smaller and the `TotRms_AbvGrd` is positive with a much larger magnitude.

```
lm(Sale_Price ~ Gr_Liv_Area, data = ames_data)$coefficients
```

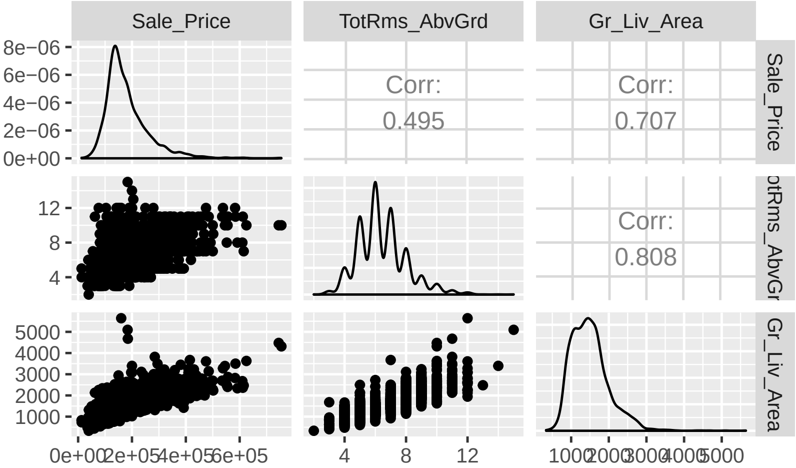
```
## (Intercept) Gr_Liv_Area
##   13289.634     111.694
```

```
lm(Sale_Price ~ TotRms_AbvGrd, data = ames_data)$coefficients
```

```
##   (Intercept) TotRms_AbvGrd
##      18665.40      25163.83
```

- This is a common result when collinearity exists.
- Coefficients for correlated features become over-inflated and can fluctuate significantly.

```r
library(GGally)
ggpairs(ames_data[,c("Sale_Price","TotRms_AbvGrd","Gr_Liv_Area")
```

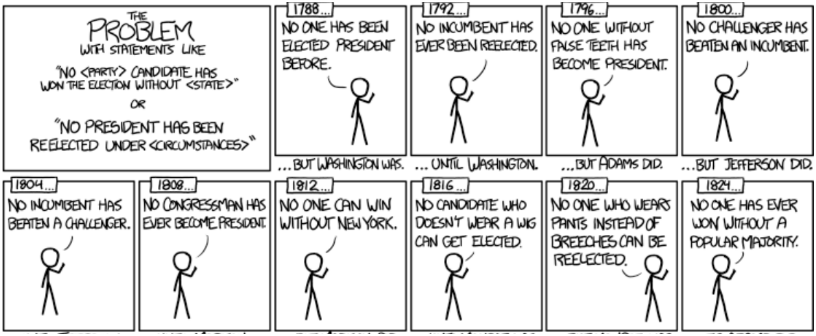


Consequences

- One consequence of these large fluctuations in the coefficient terms is **overfitting**, which means we have high variance in the bias-variance tradeoff space.
- We can use tools such as **variance inflaction factors** (Myers, 1994) to identify and remove those strongly correlated variables, but it is not always clear which variable(s) to remove.
- Nor do we always wish to remove variables as this may be removing signal in our data.

The problem - Overfitting

- Our model doesn't generalize well from our training data to unseen data.



What can be done against overvitting

- **Cross Validation**
- Train with more data
- Remove features
- **Regularization** - e.g. ridge and lasso regression
- Ensembling - e.g. bagging and boosting

Cross validation

- Cross-validation is a powerful preventative measure against overfitting.
- Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.

Cross Validation in R

Split data into training and testing dataset

```
library(caret)
library(tidyverse)
training.samples <- ames_data$Sale_Price %>%
createDataPartition(p = 0.8, list = FALSE)
train.data  <- ames_data[training.samples, ]
test.data <- ames_data[-training.samples, ]
nrow(train.data) # used to train (i.e. build) the model
## [1] 2346
nrow(test.data) # used to test (i.e. validate) the model
## [1] 584
                # by estimating the prediction error.
```

Build the model and make predictions

```
model <- lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd,
            data = train.data)
# Make predictions and compute the R2, RMSE and MAE
(predictions <- model %>% predict(test.data))
```

```
##         1         2         3         4         5         6
## 109790.24 161208.48 214778.75 124731.14 179665.14 165422.99 2
##         9        10        11        12        13        14
## 152488.80 404113.29 181381.19 251692.07 210010.81 126648.29 1
##        17        18        19        20        21        22
## 202045.65 205650.97 187512.85 122551.22 277297.69 123160.42 1
##        25        26        27        28        29        30
## 169346.85 110371.55 163824.38 149872.89 226874.76 166149.63 2
##        33        34        35        36        37        38
## 218674.72  98164.00 115256.94 230356.73 201989.88 181872.94 1
##        41        42        43        44        45        46
## 199340.19 200737.70 173561.36 172661.51 178329.30 115485.92 2
##        49        50        51        52        53        54
```

Model with cross validation

- Loocv: **leave one out cross validation**

```
train.control <- caret::trainControl(method = "LOOCV")
# Train the model
model2 <- train(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd,
                data = train.data, method = "lm",
                trControl = train.control)
model2 %>% predict(test.data)
```

Links - linear regression

- Regression - **r-bloggers**
- The complete book of **Faraway**- very intuitive
- Good introduction on **Quick-R**
- **Multiple regression**
- **15 Types of Regression you should know**
- `ggeffects` - **Create Tidy Data Frames of Marginal Effects for 'ggplot' from Model Outputs**
- **Machine learning iteration**

Shiny App - Diagnostics for linear regression

- Shiny App - **Simple Linear Regression**
- Shiny App - **Multicollinearity in multiple regression**