Machine Learning Algorithms Documentation

Geiser, Gruen, Hefti, Kuster

08/01/2021

Table of Contents

#Data preparation

**Dataset - German Housing Data**

Origin data set cleaned:

Variable rooms, bathrooms, bedrooms, floors, garages, Year\_built, Year\_renovated changed from decimal to integer Subset with buildings up to 20 rooms for our analysis created Rooms with .5 rounded up Levels of the variable Energy\_source and Garagetype transformed.

**Load of the cleaned dataset**

data <- read.csv('german\_housing\_cleaned.csv',header =T, encoding='UTF-8')  
head(data)

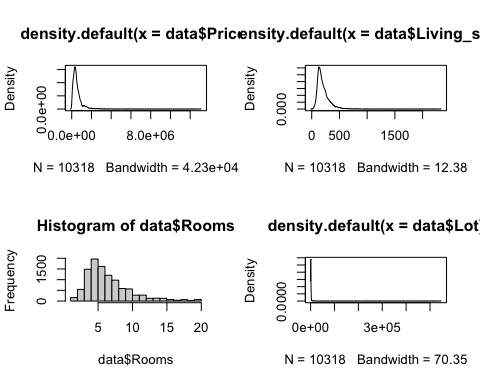
## Price Type Living\_space Lot Rooms Bedrooms Bathrooms Floors  
## 1 498000 Multiple dwelling 106.00 229 5 3 1 2  
## 2 495000 Mid-terrace house 140.93 517 6 3 2 NA  
## 3 749000 Farmhouse 162.89 82 5 3 2 4  
## 4 259000 Farmhouse 140.00 814 4 NA 2 2  
## 5 469000 Multiple dwelling 115.00 244 4 2 1 NA  
## 6 1400000 Mid-terrace house 310.00 860 8 NA NA 3  
## Year\_built Furnishing\_quality Year\_renovated Condition Heating  
## 1 2005 normal NA refurbished central heating  
## 2 1994 basic NA refurbished stove heating  
## 3 2013 NA dilapidated stove heating  
## 4 1900 basic 2000 fixer-upper central heating  
## 5 1968 refined 2019 refurbished central heating  
## 6 1969 basic NA maintained   
## Energy\_source Energy\_efficiency\_class State City  
## 1 natural gas D Baden-Württemberg Bodenseekreis  
## 2 Baden-Württemberg Konstanz (Kreis)  
## 3 district heating B Baden-Württemberg Esslingen (Kreis)  
## 4 electricity G Baden-Württemberg Waldshut (Kreis)  
## 5 oil F Baden-Württemberg Esslingen (Kreis)  
## 6 oil Baden-Württemberg Stuttgart  
## Place Garages Garagetype  
## 1 Bermatingen 2 Parking lot  
## 2 Engen 7 Parking lot  
## 3 Ostfildern 1 Garage  
## 4 Bonndorf im Schwarzwald 1 Garage  
## 5 Leinfelden-Echterdingen 1 Garage  
## 6 Süd 2 Garage

**Inspection of all variables**

str(data)  
summary(data)  
colnames(data)  
apply(data,2,function(x) sum(is.na(x)))

**Analysis of the distribution of the variables: ‘Price’, ‘Living\_space’, ‘Rooms’ and ‘Lot’**

par(mfrow = c(2,2))  
  
#Price  
plot(density(data$Price))  
  
#Living\_space  
plot(density(data$Living\_space))  
  
#Rooms  
hist(data$Rooms)  
  
#Lot  
plot(density(data$Lot))

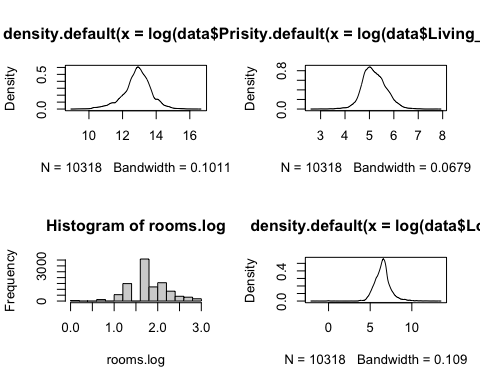


Result: The variables are right skewed

**Log Transformation of variables**

Therefore we will use the Log Transformation of the variables ‘Price’, ‘Living\_space’, ‘Rooms’, ‘Lot’ to get a nearly normal distribution

par(mfrow = c(2,2))  
  
#Price  
price.log <- density(log(data$Price))  
plot(price.log)  
  
#Living\_space  
living.log <- density(log(data$Living\_space))  
plot(living.log)  
  
#Rooms  
rooms.log <- log(data$Rooms)  
hist(rooms.log)  
  
#Lot  
lot.log <- density(log(data$Lot))  
plot(lot.log)



**Add new columns with log transformed variables price, living space and rooms**

data1 <- data  
data1$log.price <- log(data1$Price)  
data1$log.living <- log(data1$Living\_space)  
data1$log.rooms <- log(data1$Rooms)  
data1$log.lot <- log(data1$Lot)  
data1$Condition <- factor(data1$Condition)  
#str(data1)

# Week 1 - Linear Models

## Data Visualisation and Linear regressions

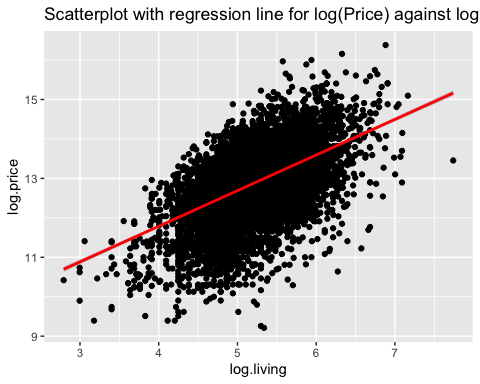
The variables Price, Living\_space and Rooms are checked for na values

options(scipen=999) #block scientific notation  
library(ggplot2)  
attach(data)

## Scatterplot with regression line for log(Price) against log(Living\_space)

We plot the response variable “Price” against the predictor “Living\_Space” to get a first impression and grahical analysis.

#Living\_space  
ggplot(data1, aes(log.living, log.price)) + geom\_point() + geom\_smooth(method = lm, se = T, color = 'red') + ggtitle('Scatterplot with regression line for log(Price) against log.living')



## Fitting a Simple Linear regression of log(Price) against log(Living\_space) and check the coefficients

#linear model  
lm.log.price\_living <- lm(log.price ~ log.living, data = data1)  
summary(lm.log.price\_living)

##   
## Call:  
## lm(formula = log.price ~ log.living, data = data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.7836 -0.4079 0.0856 0.4683 2.7581   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.17522 0.07596 107.62 <0.0000000000000002 \*\*\*  
## log.living 0.90280 0.01455 62.05 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7145 on 10316 degrees of freedom  
## Multiple R-squared: 0.2718, Adjusted R-squared: 0.2717   
## F-statistic: 3850 on 1 and 10316 DF, p-value: < 0.00000000000000022

#estimated regression coefficients  
coef(lm.log.price\_living)

## (Intercept) log.living   
## 8.1752232 0.9028032

exp(coef(lm.log.price\_living))

## (Intercept) log.living   
## 3551.847713 2.466508

#p-values  
summary(lm.log.price\_living)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 8.1752232 0.07596044 107.62475 0  
## log.living 0.9028032 0.01455063 62.04563 0

Result: very significant p-values for Price ~ Living\_space. We can assume that the variable Living\_space has an effect on the dependent variable (Price) with a positive correlation, meaning: if the Living\_space parameter increase in value also the Price of the property will increase.

We interpret the intercept and the second coefficient, the slope, we exponentiate the values. The results are: For the intercept exp(8.17522)= 3551.84 and the slope exp(0.9028)=2.47

Interpretation: It seems to be a positive relationship between these two variables. More livingspace seems to have a higher price. So we fit a simple regression model to the data. With a livingspace of 0 the price would be 3551.84 EURO and with each unit increase of the livingspace the price increase by 2.47 EURO which does not make much sense. The p-value is very small therefore we have a strong evidence that the slope for livingspace is not flat.

## Linear regression of log(Price) against log(Living\_space) including the Type and finding the intercept for the different Types.

##linear model  
lm.log.price\_living\_type <- lm(data = data1, log.price ~ log.living + Type)  
summary(lm.log.price\_living\_type)

##   
## Call:  
## lm(formula = log.price ~ log.living + Type, data = data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.9802 -0.3849 0.0716 0.4556 2.7356   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.47225 0.09409 79.414 < 0.0000000000000002 \*\*\*  
## log.living 0.99691 0.01679 59.378 < 0.0000000000000002 \*\*\*  
## TypeBungalow 0.24691 0.05737 4.304 0.000016956457415 \*\*\*  
## TypeCastle -0.33356 0.31169 -1.070 0.284573   
## TypeCorner house -0.14759 0.06058 -2.436 0.014857 \*   
## TypeDuplex -0.06320 0.03900 -1.620 0.105186   
## TypeFarmhouse 0.26026 0.04599 5.659 0.000000015603919 \*\*\*  
## TypeMid-terrace house 0.24470 0.03679 6.651 0.000000000030494 \*\*\*  
## TypeMultiple dwelling 0.35983 0.05029 7.155 0.000000000000891 \*\*\*  
## TypeResidential property 0.17411 0.05094 3.418 0.000634 \*\*\*  
## TypeSingle dwelling 0.39721 0.04091 9.708 < 0.0000000000000002 \*\*\*  
## TypeSpecial property 0.46909 0.05103 9.193 < 0.0000000000000002 \*\*\*  
## TypeVilla 0.70309 0.05077 13.847 < 0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6912 on 10305 degrees of freedom  
## Multiple R-squared: 0.3193, Adjusted R-squared: 0.3185   
## F-statistic: 402.8 on 12 and 10305 DF, p-value: < 0.00000000000000022

#estimated regression coefficients  
coef(lm.log.price\_living\_type)

## (Intercept) log.living TypeBungalow   
## 7.4722543 0.9969080 0.2469139   
## TypeCastle TypeCorner house TypeDuplex   
## -0.3335592 -0.1475928 -0.0631954   
## TypeFarmhouse TypeMid-terrace house TypeMultiple dwelling   
## 0.2602565 0.2447012 0.3598267   
## TypeResidential property TypeSingle dwelling TypeSpecial property   
## 0.1741124 0.3972086 0.4690858   
## TypeVilla   
## 0.7030881

#intercept of Type "NULL"  
coef(lm.log.price\_living\_type)['(Intercept)']

## (Intercept)   
## 7.472254

#intercept of Type single dwelling  
coef(lm.log.price\_living\_type)['(Intercept)'] + coef(lm.log.price\_living\_type)['TypeSingle dwelling']

## (Intercept)   
## 7.869463

## Linear regression of log(Price) against log(Living\_space) including the Type interaction

##linear model  
lm.log.price\_living\_type2 <- lm(data = data1, log.price ~ log.living \* Type)  
summary(lm.log.price\_living\_type2)

##   
## Call:  
## lm(formula = log.price ~ log.living \* Type, data = data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.0574 -0.3835 0.0672 0.4576 2.7139   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 7.73609 0.27923 27.705  
## log.living 0.94614 0.05331 17.749  
## TypeBungalow 0.92173 0.48628 1.895  
## TypeCastle -1.04467 5.02138 -0.208  
## TypeCorner house -0.80053 0.58144 -1.377  
## TypeDuplex 1.34421 0.35134 3.826  
## TypeFarmhouse 0.63145 0.53470 1.181  
## TypeMid-terrace house -0.68139 0.31354 -2.173  
## TypeMultiple dwelling 0.24793 0.69059 0.359  
## TypeResidential property -0.31372 0.45320 -0.692  
## TypeSingle dwelling -0.75763 0.42213 -1.795  
## TypeSpecial property -0.77518 0.45681 -1.697  
## TypeVilla 0.06219 0.60861 0.102  
## log.living:TypeBungalow -0.11718 0.08861 -1.322  
## log.living:TypeCastle 0.12151 0.79328 0.153  
## log.living:TypeCorner house 0.12396 0.10936 1.133  
## log.living:TypeDuplex -0.24967 0.06560 -3.806  
## log.living:TypeFarmhouse -0.08080 0.10831 -0.746  
## log.living:TypeMid-terrace house 0.18009 0.06011 2.996  
## log.living:TypeMultiple dwelling 0.01967 0.13964 0.141  
## log.living:TypeResidential property 0.09284 0.08538 1.087  
## log.living:TypeSingle dwelling 0.23218 0.08354 2.779  
## log.living:TypeSpecial property 0.25164 0.09095 2.767  
## log.living:TypeVilla 0.11600 0.10750 1.079  
## Pr(>|t|)   
## (Intercept) < 0.0000000000000002 \*\*\*  
## log.living < 0.0000000000000002 \*\*\*  
## TypeBungalow 0.058061 .   
## TypeCastle 0.835198   
## TypeCorner house 0.168607   
## TypeDuplex 0.000131 \*\*\*  
## TypeFarmhouse 0.237658   
## TypeMid-terrace house 0.029787 \*   
## TypeMultiple dwelling 0.719593   
## TypeResidential property 0.488813   
## TypeSingle dwelling 0.072720 .   
## TypeSpecial property 0.089737 .   
## TypeVilla 0.918609   
## log.living:TypeBungalow 0.186066   
## log.living:TypeCastle 0.878261   
## log.living:TypeCorner house 0.257038   
## log.living:TypeDuplex 0.000142 \*\*\*  
## log.living:TypeFarmhouse 0.455705   
## log.living:TypeMid-terrace house 0.002741 \*\*   
## log.living:TypeMultiple dwelling 0.887975   
## log.living:TypeResidential property 0.276883   
## log.living:TypeSingle dwelling 0.005460 \*\*   
## log.living:TypeSpecial property 0.005673 \*\*   
## log.living:TypeVilla 0.280593   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6879 on 10294 degrees of freedom  
## Multiple R-squared: 0.3264, Adjusted R-squared: 0.3249   
## F-statistic: 216.9 on 23 and 10294 DF, p-value: < 0.00000000000000022

#estimated regression coefficients  
coef(lm.log.price\_living\_type2)

## (Intercept) log.living   
## 7.73608549 0.94613962   
## TypeBungalow TypeCastle   
## 0.92172518 -1.04467031   
## TypeCorner house TypeDuplex   
## -0.80052753 1.34420839   
## TypeFarmhouse TypeMid-terrace house   
## 0.63144756 -0.68139164   
## TypeMultiple dwelling TypeResidential property   
## 0.24793247 -0.31371529   
## TypeSingle dwelling TypeSpecial property   
## -0.75763067 -0.77518375   
## TypeVilla log.living:TypeBungalow   
## 0.06219264 -0.11718366   
## log.living:TypeCastle log.living:TypeCorner house   
## 0.12151270 0.12396267   
## log.living:TypeDuplex log.living:TypeFarmhouse   
## -0.24967456 -0.08079852   
## log.living:TypeMid-terrace house log.living:TypeMultiple dwelling   
## 0.18008764 0.01967192   
## log.living:TypeResidential property log.living:TypeSingle dwelling   
## 0.09283769 0.23217929   
## log.living:TypeSpecial property log.living:TypeVilla   
## 0.25164379 0.11599769

# The "P-Values - confidence intervals" duality  
confint(lm.log.price\_living\_type2)

## 2.5 % 97.5 %  
## (Intercept) 7.18873463 8.28343635  
## log.living 0.84164786 1.05063138  
## TypeBungalow -0.03148560 1.87493595  
## TypeCastle -10.88754235 8.79820173  
## TypeCorner house -1.94027217 0.33921710  
## TypeDuplex 0.65550987 2.03290692  
## TypeFarmhouse -0.41667545 1.67957057  
## TypeMid-terrace house -1.29599299 -0.06679028  
## TypeMultiple dwelling -1.10576712 1.60163207  
## TypeResidential property -1.20207438 0.57464379  
## TypeSingle dwelling -1.58509414 0.06983281  
## TypeSpecial property -1.67062159 0.12025410  
## TypeVilla -1.13079934 1.25518461  
## log.living:TypeBungalow -0.29088557 0.05651824  
## log.living:TypeCastle -1.43346849 1.67649390  
## log.living:TypeCorner house -0.09041307 0.33833842  
## log.living:TypeDuplex -0.37826357 -0.12108555  
## log.living:TypeFarmhouse -0.29311479 0.13151774  
## log.living:TypeMid-terrace house 0.06226578 0.29790949  
## log.living:TypeMultiple dwelling -0.25405926 0.29340310  
## log.living:TypeResidential property -0.07451478 0.26019016  
## log.living:TypeSingle dwelling 0.06841797 0.39594061  
## log.living:TypeSpecial property 0.07335498 0.42993261  
## log.living:TypeVilla -0.09472399 0.32671936

Result: mostly very significant p-values for Price ~ Living\_space and different types. Only TypeCastle and TypeDuplex with non significant values. We can assume that the different types have different impact on the variable Price against Living\_space.

Interpretation of the coefficients of Type variable: Intercept for TypeSingle Dwelling: exp(7.869463) with Slope: exp(0.9969080)

### Measures of fit

formula(lm.log.price\_living\_type)

## log.price ~ log.living + Type

#r.squared  
summary(lm.log.price\_living\_type)$r.squared

## [1] 0.3192727

#adj.r.squared  
summary(lm.log.price\_living\_type)$adj.r.squared

## [1] 0.31848

formula(lm.log.price\_living\_type2)

## log.price ~ log.living \* Type

#r.squared  
summary(lm.log.price\_living\_type2)$r.squared

## [1] 0.3263949

#adj.r.squared  
summary(lm.log.price\_living\_type2)$adj.r.squared

## [1] 0.3248899

## Linear regression of log(Price) against log(Rooms)

In an next step we examine the data set graphically and consider again “Price” as response variable but as predictor “Rooms” and we fit the model again with a simple linear regression.

### Measures of fit

formula(lm.log.price\_rooms)

## log(Price) ~ log(Rooms)

#r.squared  
summary(lm.log.price\_rooms)$r.squared

## [1] 0.09968285

#adj.r.squared  
summary(lm.log.price\_rooms)$adj.r.squared

## [1] 0.09959558

## Linear regression of log(Price) against Type

Now we are modelling a linear regression with dependent variable ‘Price’ and the categorical variable “Type” as independent variable. First we visualize the variable ‘Type’ with a boxplot with and one without outliers.

## Fitted values

**The function fitted() can be used to extract the predicted values for the existing observations**

attach(data)

## The following objects are masked from data (pos = 3):  
##   
## Bathrooms, Bedrooms, City, Condition, Energy\_efficiency\_class,  
## Energy\_source, Floors, Furnishing\_quality, Garages, Garagetype,  
## Heating, Living\_space, Lot, Place, Price, Rooms, State, Type,  
## Year\_built, Year\_renovated

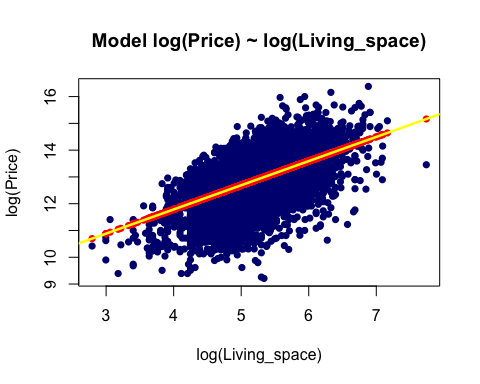
#lm.log.price\_living  
fitted.price\_living <- fitted(lm.log.price\_living)  
str(fitted.price\_living)

## Named num [1:10318] 12.4 12.6 12.8 12.6 12.5 ...  
## - attr(\*, "names")= chr [1:10318] "1" "2" "3" "4" ...

head(fitted.price\_living)

## 1 2 3 4 5 6   
## 12.38539 12.64253 12.77327 12.63655 12.45896 13.35422

plot(log(Price)~ log(Living\_space), main = 'Model log(Price) ~ log(Living\_space)', col = 'navy', pch = 16)  
points(fitted.price\_living ~ log(Living\_space), col = 'red', pch = 16)  
abline(lm.log.price\_living, col = 'yellow', lwd = 2.5)



## Residuals of model log(Price) ~ log(Living\_space)

attach(data1)  
resid.price\_living <- resid(lm.log.price\_living)  
length(resid.price\_living)

## [1] 10318

head(resid.price\_living)

## 1 2 3 4 5 6   
## 0.7329643 0.4697818 0.7532264 -0.1719706 0.5993948 0.7977636

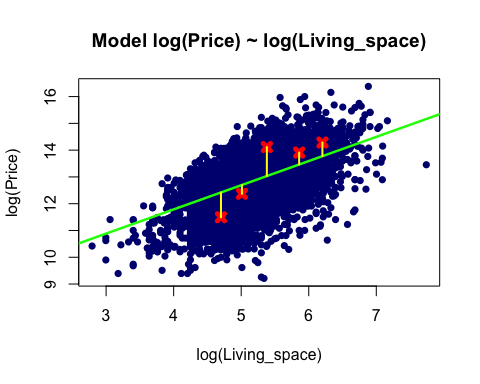
set.seed(100)  
id <- sample(x = 1:10318, size = 5)  
resid.price\_living[id]

## 3786 503 3430 3696 4090   
## 0.5106768 -0.3315001 0.4470366 -0.9261093 1.0834033

fitted.price\_living[id]

## 3786 503 3430 3696 4090   
## 13.77484 12.69884 13.46378 12.41883 13.03221

plot(log(Price) ~ log(Living\_space), main = 'Model log(Price) ~ log(Living\_space)', col = 'navy', pch = 16)  
abline(lm.log.price\_living, col = 'green', lwd = 2.5)  
  
points(log(Price) ~ log(Living\_space), data = data1[id, ], col = 'red', pch = 4, lwd = 5)  
segments(x0 = data1[id, 'log.living'], x1 = data1[id, 'log.living'],  
 y0 = fitted.price\_living[id], y1 = data1[id, 'log.price'], col = 'yellow', lwd = 2)



## Predicting values using splitted data set 80:20 ratio

#split dataset   
split80 <- round(nrow(data1)\* 0.80)  
train <- data1[1:split80,]  
test <- data1[(split80 + 1):nrow(data1),]  
dim(train)

## [1] 8254 24

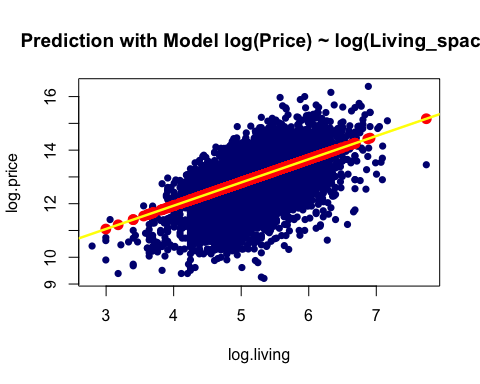
dim(test)

## [1] 2064 24

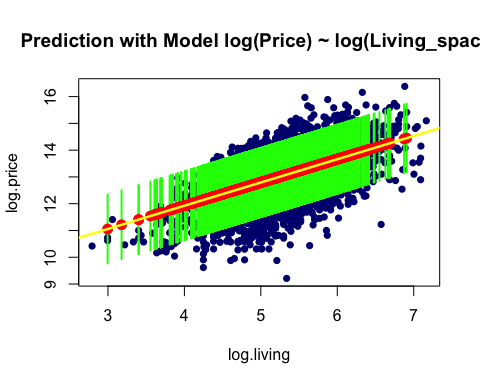
#linear regression model  
lm.train <- lm(log.price ~ log.living, data = train)  
summary(lm.train)

##   
## Call:  
## lm(formula = log.price ~ log.living, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.8765 -0.3774 0.0557 0.4283 2.6734   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.45284 0.07981 105.91 <0.0000000000000002 \*\*\*  
## log.living 0.86819 0.01524 56.96 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6563 on 8252 degrees of freedom  
## Multiple R-squared: 0.2822, Adjusted R-squared: 0.2821   
## F-statistic: 3245 on 1 and 8252 DF, p-value: < 0.00000000000000022

#predictions  
pred.new.living <- predict(object = lm.train, newdata = test)  
pred.new.living.CI <- predict(object = lm.train, interval = 'prediction', newdata = test)  
  
#display predictions  
plot(log.price ~log.living, data = data1, main = 'Prediction with Model log(Price) ~ log(Living\_space)', col = 'navy', pch = 16)  
points(x = test$log.living, y= pred.new.living, col = 'red', pch = 16, cex = 1.5)  
abline(lm.train, col = 'yellow', lwd = 2.5)

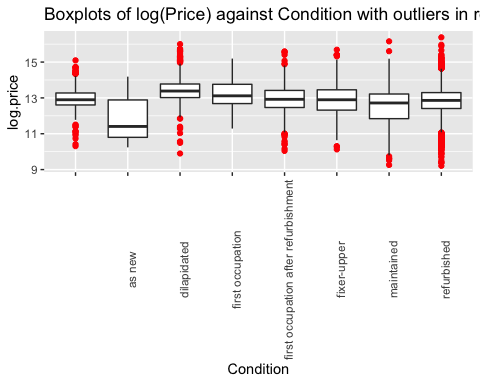


plot(log.price ~ log.living, data = train, main = 'Prediction with Model log(Price) ~ log(Living\_space)', col = 'navy', pch = 16)  
segments(x0 = test$log.living, x1 = test$log.living,  
 y0 = pred.new.living.CI[, 'lwr'], y1 = pred.new.living.CI[, 'upr'], lwd = 2, col = 'green')  
points(x = test$log.living, y= pred.new.living.CI[,'fit'], col = 'red', pch = 16, cex =1.5)  
abline(lm.train, col = 'yellow', lwd = 2.5)



## Testing the effect of a categorical variable and post-hoc contrasts

condition.box.with\_outlier <- ggplot(data1, aes(x=Condition, y=log.price)) + geom\_boxplot(outlier.colour = 'red')+ theme(axis.text.x = element\_text(angle = 90)) + ggtitle('Boxplots of log(Price) against Condition with outliers in red')  
plot(condition.box.with\_outlier)



#model  
lm.price\_condition.1 <- lm(log.price ~ Condition, data = data1)  
summary(lm.price\_condition.1)

##   
## Call:  
## lm(formula = log.price ~ Condition, data = data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.6043 -0.4304 0.0333 0.4884 3.6247   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 12.95695 0.04213 307.553  
## Conditionas new -1.01885 0.24696 -4.126  
## Conditiondilapidated 0.47108 0.04838 9.738  
## Conditionfirst occupation 0.19331 0.09330 2.072  
## Conditionfirst occupation after refurbishment -0.05091 0.05685 -0.895  
## Conditionfixer-upper -0.10881 0.05398 -2.016  
## Conditionmaintained -0.42435 0.04906 -8.649  
## Conditionrefurbished -0.14229 0.04328 -3.288  
## Pr(>|t|)   
## (Intercept) < 0.0000000000000002 \*\*\*  
## Conditionas new 0.0000373 \*\*\*  
## Conditiondilapidated < 0.0000000000000002 \*\*\*  
## Conditionfirst occupation 0.03829 \*   
## Conditionfirst occupation after refurbishment 0.37055   
## Conditionfixer-upper 0.04384 \*   
## Conditionmaintained < 0.0000000000000002 \*\*\*  
## Conditionrefurbished 0.00101 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8071 on 10310 degrees of freedom  
## Multiple R-squared: 0.07146, Adjusted R-squared: 0.07083   
## F-statistic: 113.4 on 7 and 10310 DF, p-value: < 0.00000000000000022

#coefficients  
coef(lm.price\_condition.1)

## (Intercept)   
## 12.95694626   
## Conditionas new   
## -1.01884973   
## Conditiondilapidated   
## 0.47108462   
## Conditionfirst occupation   
## 0.19330741   
## Conditionfirst occupation after refurbishment   
## -0.05090966   
## Conditionfixer-upper   
## -0.10881029   
## Conditionmaintained   
## -0.42434950   
## Conditionrefurbished   
## -0.14228677

aggregate(log.price ~Condition,   
 FUN = mean, data = data1)

## Condition log.price  
## 1 12.95695  
## 2 as new 11.93810  
## 3 dilapidated 13.42803  
## 4 first occupation 13.15025  
## 5 first occupation after refurbishment 12.90604  
## 6 fixer-upper 12.84814  
## 7 maintained 12.53260  
## 8 refurbished 12.81466

#model without slope, only intercept  
lm.price\_condition.0 <- lm(log.price ~ 1, data = data1)  
summary(lm.price\_condition.0)

##   
## Call:  
## lm(formula = log.price ~ 1, data = data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.6576 -0.4388 0.0287 0.5166 3.5125   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 12.867983 0.008243 1561 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8373 on 10317 degrees of freedom

coef(lm.price\_condition.0)

## (Intercept)   
## 12.86798

#Anova  
anova(lm.price\_condition.0, lm.price\_condition.1)

## Analysis of Variance Table  
##   
## Model 1: log.price ~ 1  
## Model 2: log.price ~ Condition  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 10317 7232.5   
## 2 10310 6715.7 7 516.86 113.36 < 0.00000000000000022 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#post-hoc contrasts  
library(multcomp)  
unique(data1$Condition)

## [1] refurbished dilapidated   
## [3] fixer-upper maintained   
## [5] as new   
## [7] first occupation after refurbishment first occupation   
## 8 Levels: as new dilapidated ... refurbished

ph.test.1 <- glht(model = lm.price\_condition.1, linfct = mcp(Condition = c('refurbished - dilapidated = 0')))  
summary(ph.test.1)

##   
## Simultaneous Tests for General Linear Hypotheses  
##   
## Multiple Comparisons of Means: User-defined Contrasts  
##   
##   
## Fit: lm(formula = log.price ~ Condition, data = data1)  
##   
## Linear Hypotheses:  
## Estimate Std. Error t value Pr(>|t|)  
## refurbished - dilapidated == 0 -0.61337 0.02576 -23.81 <0.0000000000000002  
##   
## refurbished - dilapidated == 0 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## (Adjusted p values reported -- single-step method)

R uses “treatment contrasts” and therefore the Intercept refers to the first in alphabetical order, here “Null”. The other coefficients represent the difference.

## Adding more categorical variables to the testing above

#str(data1)  
Year\_built1 <- as.integer(data1$Year\_built)  
floor1 <- as.integer(data1$Floors)  
typeof(Year\_built1)

## [1] "integer"

lm.price\_condition.2 <- update(lm.price\_condition.1,. ~ . + Type + log.rooms + State +Energy\_efficiency\_class + Year\_built1 + Furnishing\_quality + floor1)  
formula(lm.price\_condition.2)

## log.price ~ Condition + Type + log.rooms + State + Energy\_efficiency\_class +   
## Year\_built1 + Furnishing\_quality + floor1

drop1(lm.price\_condition.2, test = "F")

## Single term deletions  
##   
## Model:  
## log.price ~ Condition + Type + log.rooms + State + Energy\_efficiency\_class +   
## Year\_built1 + Furnishing\_quality + floor1  
## Df Sum of Sq RSS AIC F value  
## <none> 2148.4 -8631.8   
## Condition 7 17.57 2165.9 -8587.1 8.3684  
## Type 11 134.63 2283.0 -8215.5 40.7950  
## log.rooms 1 152.62 2301.0 -8138.9 508.7221  
## State 15 638.74 2787.1 -6784.8 141.9381  
## Energy\_efficiency\_class 9 33.56 2181.9 -8538.0 12.4294  
## Year\_built1 1 56.88 2205.2 -8445.4 189.5786  
## Furnishing\_quality 4 346.08 2494.4 -7562.8 288.3902  
## floor1 1 31.86 2180.2 -8527.7 106.1896  
## Pr(>F)   
## <none>   
## Condition 0.0000000003205 \*\*\*  
## Type < 0.00000000000000022 \*\*\*  
## log.rooms < 0.00000000000000022 \*\*\*  
## State < 0.00000000000000022 \*\*\*  
## Energy\_efficiency\_class < 0.00000000000000022 \*\*\*  
## Year\_built1 < 0.00000000000000022 \*\*\*  
## Furnishing\_quality < 0.00000000000000022 \*\*\*  
## floor1 < 0.00000000000000022 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Testing categorical variable (Furnishing\_quality)and comparing by F-test (Soll das hier noch rein ??)

# Week 2 - Non-linearity

## Polynomials

By including polynomials (e.g. x1 + x1^2) we can model non linear relationships with a Linear Model.

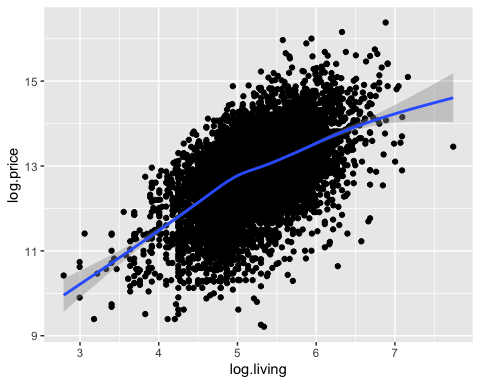
library(ggplot2)  
attach(data1)

**Graphical analysis**

log(Price) ~ log(Living\_space)

gg.log.price\_log.living <- ggplot(data1,mapping = aes(y = log.price, x = log.living)) + geom\_point()  
gg.log.price\_log.living + geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

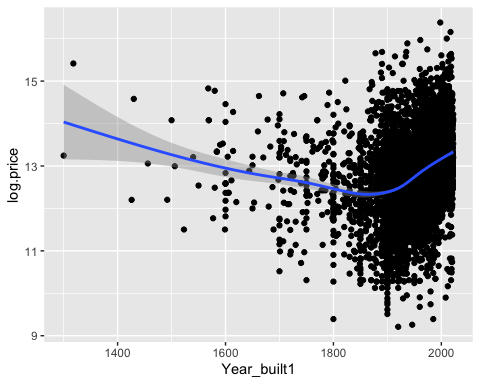


gg.log.price\_log.living <- ggplot(data1,mapping = aes(y = log.price, x = Year\_built1)) + geom\_point()  
gg.log.price\_log.living + geom\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 666 rows containing non-finite values (stat\_smooth).

## Warning: Removed 666 rows containing missing values (geom\_point).



Quadratic Effects

#model with a linear effect for log.living  
lm.living.1 <- lm(log.price ~ log.living + Year\_built1)  
summary(lm.living.1)

##   
## Call:  
## lm(formula = log.price ~ log.living + Year\_built1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.6576 -0.3852 0.0262 0.4060 3.9425   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.2750992 0.2533033 -5.034 0.000000489 \*\*\*  
## log.living 0.9446621 0.0139966 67.492 < 0.0000000000000002 \*\*\*  
## Year\_built1 0.0047208 0.0001203 39.234 < 0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6523 on 9649 degrees of freedom  
## (666 observations deleted due to missingness)  
## Multiple R-squared: 0.369, Adjusted R-squared: 0.3689   
## F-statistic: 2822 on 2 and 9649 DF, p-value: < 0.00000000000000022

#model with a quadratic effect for log.living  
lm.living.2 <- update(lm.living.1, . ~ . + I(log.living^2))  
summary(lm.living.2)

##   
## Call:  
## lm(formula = log.price ~ log.living + Year\_built1 + I(log.living^2))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.6813 -0.3840 0.0215 0.4053 4.1527   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -3.8930634 0.5350769 -7.276 0.000000000000371 \*\*\*  
## log.living 1.9999960 0.1905934 10.494 < 0.0000000000000002 \*\*\*  
## Year\_built1 0.0046546 0.0001207 38.553 < 0.0000000000000002 \*\*\*  
## I(log.living^2) -0.1004879 0.0180993 -5.552 0.000000028983449 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.6513 on 9648 degrees of freedom  
## (666 observations deleted due to missingness)  
## Multiple R-squared: 0.3711, Adjusted R-squared: 0.3709   
## F-statistic: 1897 on 3 and 9648 DF, p-value: < 0.00000000000000022

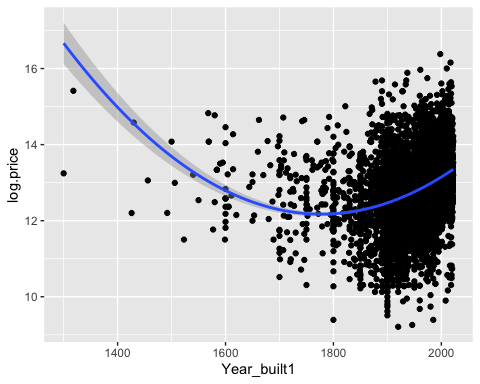
#test in quadratic  
 anova(lm.living.1, lm.living.2)

## Analysis of Variance Table  
##   
## Model 1: log.price ~ log.living + Year\_built1  
## Model 2: log.price ~ log.living + Year\_built1 + I(log.living^2)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 9649 4106.2   
## 2 9648 4093.1 1 13.077 30.825 0.00000002898 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#plot  
gg.log.price\_log.living + geom\_smooth(method = 'lm', formula = y ~poly(x, degree = 2))

## Warning: Removed 666 rows containing non-finite values (stat\_smooth).

## Warning: Removed 666 rows containing missing values (geom\_point).



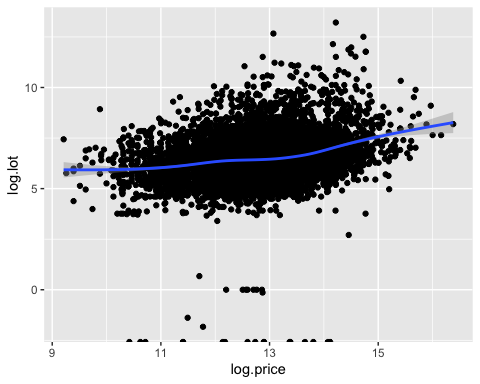
#model with a quadratic poly  
lm.living.3 <- lm(log.price ~ log.rooms + log.living + poly(log.living, degree = 2))  
summary(lm.living.3)

##   
## Call:  
## lm(formula = log.price ~ log.rooms + log.living + poly(log.living,   
## degree = 2))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.9761 -0.3927 0.0714 0.4552 2.6652   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 7.36089 0.08734 84.28 <0.0000000000000002  
## log.rooms -0.44504 0.02490 -17.88 <0.0000000000000002  
## log.living 1.21643 0.02262 53.78 <0.0000000000000002  
## poly(log.living, degree = 2)1 NA NA NA NA  
## poly(log.living, degree = 2)2 -7.82809 0.70453 -11.11 <0.0000000000000002  
##   
## (Intercept) \*\*\*  
## log.rooms \*\*\*  
## log.living \*\*\*  
## poly(log.living, degree = 2)1   
## poly(log.living, degree = 2)2 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7009 on 10314 degrees of freedom  
## Multiple R-squared: 0.2994, Adjusted R-squared: 0.2992   
## F-statistic: 1469 on 3 and 10314 DF, p-value: < 0.00000000000000022

#model with a cubic poly  
lm.living.4 <- lm(log.price ~ log.rooms + log.living + poly(log.living, degree = 3))  
summary(lm.living.4)

##   
## Call:  
## lm(formula = log.price ~ log.rooms + log.living + poly(log.living,   
## degree = 3))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.9763 -0.3928 0.0714 0.4553 2.6650   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 7.36062 0.08761 84.011 <0.0000000000000002  
## log.rooms -0.44519 0.02519 -17.676 <0.0000000000000002  
## log.living 1.21654 0.02278 53.412 <0.0000000000000002  
## poly(log.living, degree = 3)1 NA NA NA NA  
## poly(log.living, degree = 3)2 -7.82850 0.70465 -11.110 <0.0000000000000002  
## poly(log.living, degree = 3)3 -0.02715 0.70903 -0.038 0.969  
##   
## (Intercept) \*\*\*  
## log.rooms \*\*\*  
## log.living \*\*\*  
## poly(log.living, degree = 3)1   
## poly(log.living, degree = 3)2 \*\*\*  
## poly(log.living, degree = 3)3   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.701 on 10313 degrees of freedom  
## Multiple R-squared: 0.2994, Adjusted R-squared: 0.2991   
## F-statistic: 1102 on 4 and 10313 DF, p-value: < 0.00000000000000022

gg.log.lot.log.price <- ggplot(data = data1, mapping = aes(y = log.lot, x = log.price)) + geom\_point()  
gg.log.lot.log.price + geom\_smooth(method = 'gam')



## Regression Splines

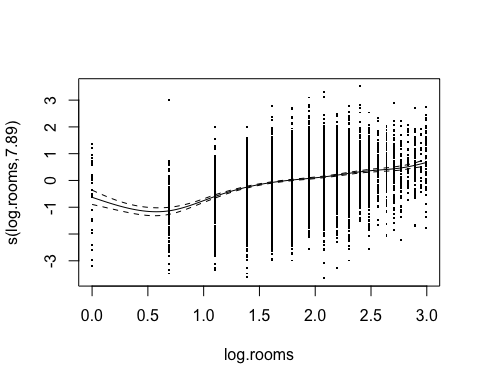
## Generalised Additive Models - GAMs

GAMs for log(Price) ~ log(Rooms)

gam.log.price.log.rooms <- gam(log.price ~ s(log.rooms))  
summary(gam.log.price.log.rooms)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## log.price ~ s(log.rooms)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 12.867983 0.007783 1653 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(log.rooms) 7.888 8.623 145.8 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.108 Deviance explained = 10.9%  
## GCV = 0.62562 Scale est. = 0.62509 n = 10318

plot(gam.log.price.log.rooms, residuals = TRUE, cex = 2)

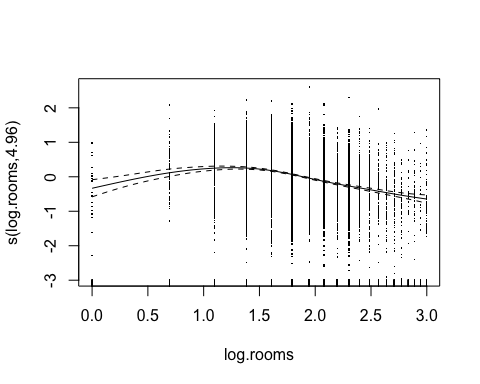


**GAMs for log(Price) ~ log(Living\_space) + s(Log(Rooms)) + s(log(Garages)**

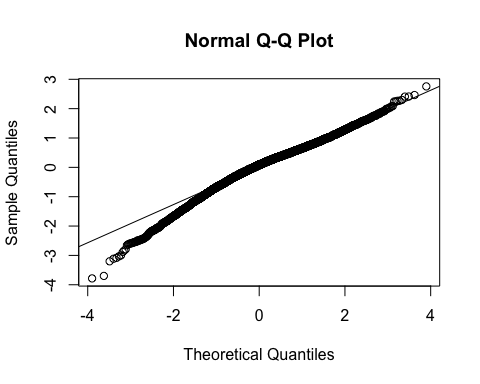
gam.log.price.log.living <- gam(log.price ~ log.living + s(log.rooms) + s(log(Garages)))  
summary(gam.log.price.log.living)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## log.price ~ log.living + s(log.rooms) + s(log(Garages))  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.46447 0.13313 48.56 <0.0000000000000002 \*\*\*  
## log.living 1.24132 0.02547 48.74 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(log.rooms) 4.964 6.084 67.976 < 0.0000000000000002 \*\*\*  
## s(log(Garages)) 8.215 8.681 3.852 0.0000713 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.305 Deviance explained = 30.6%  
## GCV = 0.42626 Scale est. = 0.4255 n = 8437

plot(gam.log.price.log.living, residuals = TRUE, select = 1)



#stroe the residuals in data1  
data1$resid.price\_living <- resid(lm.log.price\_living)  
  
#creat QQ-Plot  
  
qqnorm(resid(lm.log.price\_living))  
qqline(resid(lm.log.price\_living))

 **Expected value of the errors is “on zero”**

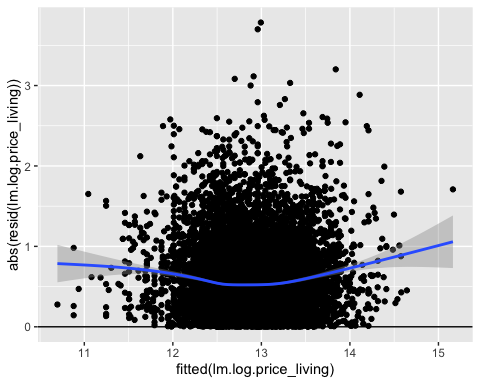
ggplot(mapping = aes(y = resid(lm.log.price\_living),  
 x = fitted(lm.log.price\_living))) +  
 geom\_abline(intercept = 0, slope = 0) + geom\_point() +  
 geom\_smooth()

probably non-linear because the smoother is not on zero

**Homoscedasticity**

ggplot(mapping = aes(y = abs(resid(lm.log.price\_living)), x = fitted(lm.log.price\_living))) +  
 geom\_abline(intercept = 0, slope = 0) + geom\_point() +  
 geom\_smooth()

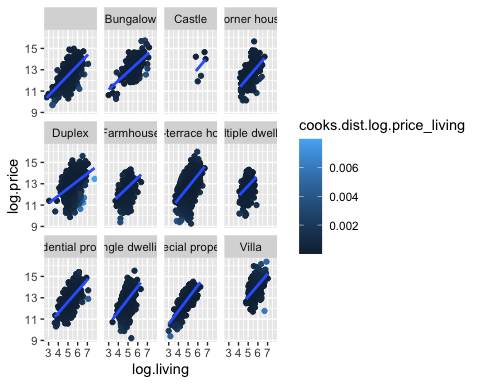
## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

 The variance of the residuals seems to be fairly constant

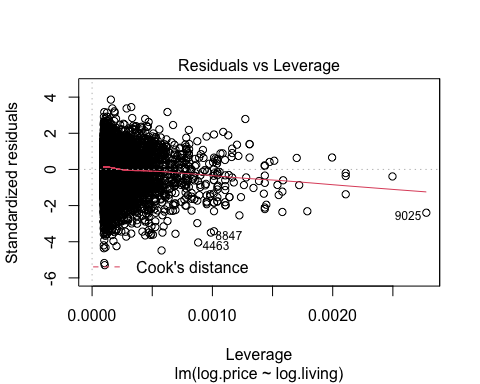
**Cooks distance**

## 1) compute Cook’s distance   
  
cooks.dist.log.price\_living <- cooks.distance(lm.log.price\_living)   
  
##  
## 2) plot it  
  
ggplot(data = data1,  
 mapping = aes(y = log.price, x = log.living,  
 colour = cooks.dist.log.price\_living)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = FALSE) + facet\_wrap(. ~ Type)

## `geom\_smooth()` using formula 'y ~ x'

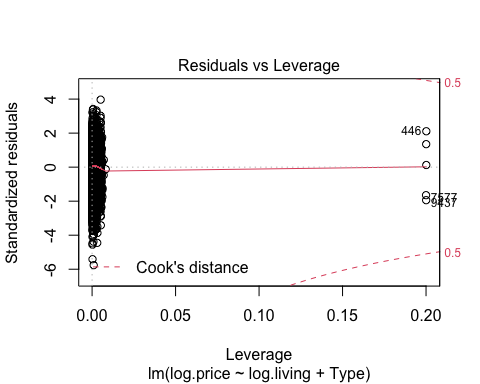
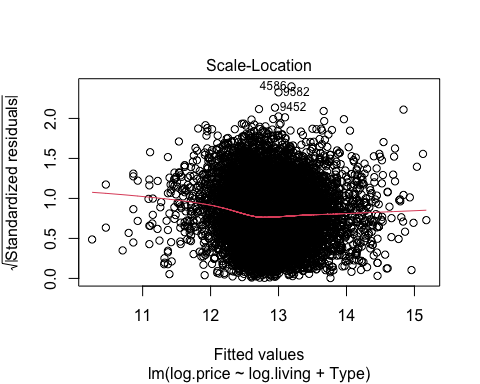
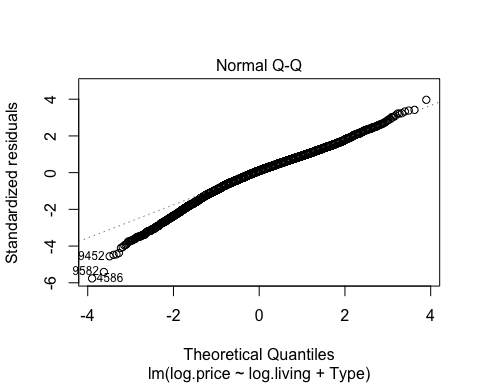
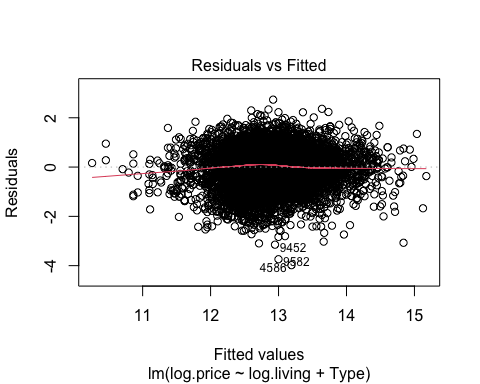


plot(lm.log.price\_living, which = 5)



**Testing the model assumptions**

plot(lm.log.price\_living\_type)

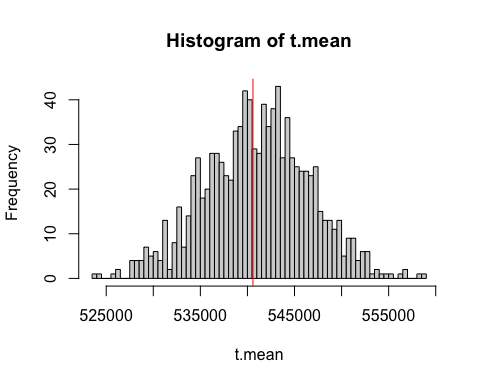


**Bootstrap**

#mean(data$Price)  
##  
B <- 10^3  
t.mean <- c()  
  
for(i in 1:B){  
 t.id <- sample(1:10318, replace = TRUE)  
 t.data.price <- data$Price[t.id]  
 t.mean[i] <- mean(t.data.price)  
}  
  
  
##  
length(t.mean)

## [1] 1000

hist(t.mean, breaks = 50)  
abline(v = mean(data$Price), col = "red")



sorted.means <- sort(t.mean)   
quantile(sorted.means, probs = c(0.025, 0.975))

## 2.5% 97.5%   
## 529609.2 551910.5

# Week 3 - Generalised Linear Models

## GLM - Possion Model

**Count Data**

With the GLM function and the family “possion” we could generalize the Linear model that the right-skewed data Rooms and Type can be followed without log- transformation in the first place.

#str(data)  
glm.rooms <- glm(Rooms ~ Type,  
family = "poisson",  
data = data)  
  
summary(glm.rooms)

##   
## Call:  
## glm(formula = Rooms ~ Type, family = "poisson", data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.6463 -0.6320 -0.0781 0.3702 4.4559   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 2.02917 0.01848 109.821 < 0.0000000000000002 \*\*\*  
## TypeBungalow 0.17457 0.02838 6.152 0.000000000766 \*\*\*  
## TypeCastle 0.55104 0.12447 4.427 0.000009550749 \*\*\*  
## TypeCorner house 0.01343 0.03162 0.425 0.671   
## TypeDuplex 0.26787 0.01981 13.522 < 0.0000000000000002 \*\*\*  
## TypeFarmhouse -0.43417 0.02638 -16.456 < 0.0000000000000002 \*\*\*  
## TypeMid-terrace house -0.22653 0.01948 -11.626 < 0.0000000000000002 \*\*\*  
## TypeMultiple dwelling -0.38499 0.02917 -13.197 < 0.0000000000000002 \*\*\*  
## TypeResidential property 0.04744 0.02637 1.799 0.072 .   
## TypeSingle dwelling -0.38432 0.02252 -17.066 < 0.0000000000000002 \*\*\*  
## TypeSpecial property -0.47841 0.03056 -15.655 < 0.0000000000000002 \*\*\*  
## TypeVilla 0.14646 0.02522 5.806 0.000000006387 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 13951.1 on 10317 degrees of freedom  
## Residual deviance: 9350.9 on 10306 degrees of freedom  
## AIC: 47557  
##   
## Number of Fisher Scoring iterations: 4

#exponential function, is needed duelog() function as link function within glm  
exp(coef(glm.rooms))

## (Intercept) TypeBungalow TypeCastle   
## 7.6077922 1.1907296 1.7350632   
## TypeCorner house TypeDuplex TypeFarmhouse   
## 1.0135214 1.3071727 0.6477992   
## TypeMid-terrace house TypeMultiple dwelling TypeResidential property   
## 0.7972979 0.6804577 1.0485829   
## TypeSingle dwelling TypeSpecial property TypeVilla   
## 0.6809147 0.6197703 1.1577330

Before the Interpretation of the coefficients of a Poisson model the inverse, the exponential, is needed due to use of the link function (log() function). The Interpretation of the coefficients are: A house with no Type information has around 7.6 rooms and e.g. Villa has ca. 15.77% more rooms (ca. 8.8 rooms)

#to double check single house (type) is checked and predicted  
  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ tibble 3.0.4 ✓ dplyr 1.0.2  
## ✓ tidyr 1.1.2 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.0  
## ✓ purrr 0.3.4

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::collapse() masks nlme::collapse()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::select() masks MASS::select()

#which(data$Type == "")  
  
#data[99,]  
fitted.room <- fitted(glm.rooms)[99]  
fitted.room

## 99   
## 7.607792

specific.room <- data[99,]  
specific.room$Type <- "Villa"  
#specific.room  
  
pred.specific.room <- predict(glm.rooms,  
 type = "response",  
 newdata = specific.room)  
pred.specific.room

## 99   
## 8.807792

fitted.room \* exp(coef(glm.rooms)["TypeVilla"])

## 99   
## 8.807792

This house with no specific Type, according to the model, is expected to have 7.607 Rooms. If we check the number of rooms after setting the Type to e.g. Villa the model expect 8.807 rooms and this is exactly the same number of rooms we get for the fitted room times the exponential estimated for Villa

**data simulation from the glm count data model**

#data simulation from the glm model (glm.rooms)  
set.seed(99)  
sim.data.rooms.Poisson <- simulate(glm.rooms)  
##  
NROW(sim.data.rooms.Poisson)

## [1] 10318

head(sim.data.rooms.Poisson)

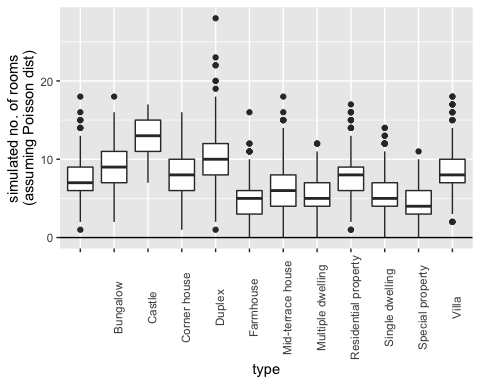
## sim\_1  
## 1 5  
## 2 3  
## 3 6  
## 4 11  
## 5 5  
## 6 11

tail(sim.data.rooms.Poisson)

## sim\_1  
## 10313 5  
## 10314 4  
## 10315 6  
## 10316 10  
## 10317 6  
## 10318 4

**Visualization of the glm count data model**

ggplot(mapping = aes(y = sim.data.rooms.Poisson$sim\_1,  
 x = data$Type)) +  
 geom\_boxplot() +  
 geom\_hline(yintercept = 0) +  
 ylab("simulated no. of rooms\n(assuming Poisson dist)") +  
 xlab("type") + theme(axis.text.x = element\_text(angle = 90))

 The results of the simulation seem to agree with the observation, see for example Type = Villa, but there are some difference due to outliers

**GLM with binomial data continious variable**

df.glm <- data.frame(data$Price, data$Living\_space, data$Year\_built)  
  
df.glm <-na.omit(df.glm)  
  
any(is.na(df.glm))

## [1] FALSE

str(df.glm)

## 'data.frame': 9652 obs. of 3 variables:  
## $ data.Price : num 498000 495000 749000 259000 469000 1400000 3500000 630000 364000 1750000 ...  
## $ data.Living\_space: num 106 141 163 140 115 ...  
## $ data.Year\_built : num 2005 1994 2013 1900 1968 ...  
## - attr(\*, "na.action")= 'omit' Named int [1:666] 29 72 90 91 117 169 173 174 181 182 ...  
## ..- attr(\*, "names")= chr [1:666] "29" "72" "90" "91" ...

glm.sq.price <- glm(cbind(data.Price, data.Living\_space)~ data.Year\_built,  
 family = "binomial",  
 data = df.glm)

## Warning in eval(family$initialize): non-integer counts in a binomial glm!

summary(glm.sq.price)

##   
## Call:  
## glm(formula = cbind(data.Price, data.Living\_space) ~ data.Year\_built,   
## family = "binomial", data = df.glm)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -74.091 -7.312 -1.783 3.161 97.560   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.500558303 0.017151213 204.1 <0.0000000000000002 \*\*\*  
## data.Year\_built 0.002238126 0.000008763 255.4 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 947369 on 9651 degrees of freedom  
## Residual deviance: 895142 on 9650 degrees of freedom  
## AIC: 963053  
##   
## Number of Fisher Scoring iterations: 6

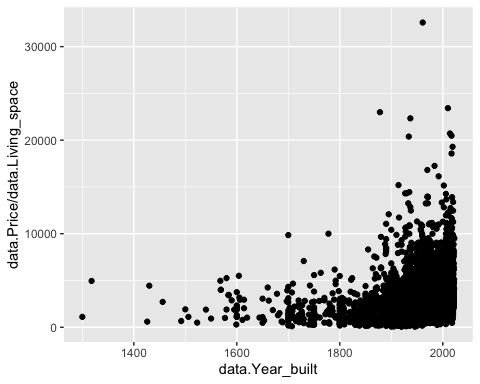
exp(coef(glm.sq.price))

## (Intercept) data.Year\_built   
## 33.133946 1.002241

ggplot(data = df.glm,  
 mapping = aes(y = data.Price/data.Living\_space,  
 x = data.Year\_built)) +   
 geom\_point() +  
 geom\_smooth(method = "glm",   
 se = FALSE,  
 method.args = list(family = "binomial"))

## `geom\_smooth()` using formula 'y ~ x'

## Warning: Computation failed in `stat\_smooth()`:  
## y values must be 0 <= y <= 1

 **GLM with binomial data factor variable**

glm.sq.price <- glm(cbind(Price, Living\_space)~ State,  
 family = "binomial",  
 data = data)

## Warning in eval(family$initialize): non-integer counts in a binomial glm!

summary(glm.sq.price)

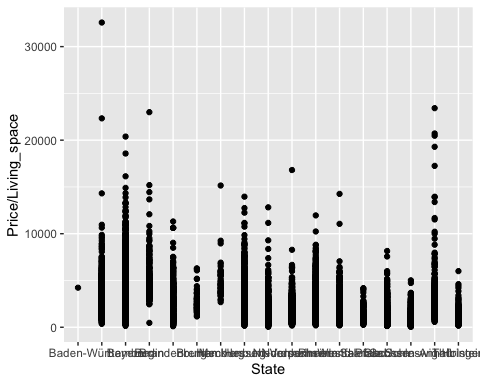
##   
## Call:  
## glm(formula = cbind(Price, Living\_space) ~ State, family = "binomial",   
## data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -59.255 -6.366 -1.195 3.473 61.109   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 8.34931 0.09536 87.558 < 0.0000000000000002  
## StateBaden-Württemberg -0.26983 0.09538 -2.829 0.004667  
## StateBayern -0.11439 0.09538 -1.199 0.230381  
## StateBerlin 0.22109 0.09549 2.315 0.020597  
## StateBrandenburg -0.34169 0.09542 -3.581 0.000342  
## StateBremen -0.45021 0.09601 -4.689 0.000002744267682839  
## StateHamburg 0.20992 0.09607 2.185 0.028878  
## StateHessen -0.35562 0.09538 -3.728 0.000193  
## StateMecklenburg-Vorpommern -0.77292 0.09544 -8.099 0.000000000000000555  
## StateNiedersachsen -0.78534 0.09538 -8.234 < 0.0000000000000002  
## StateNordrhein-Westfalen -0.52038 0.09537 -5.456 0.000000048613961013  
## StateRheinland-Pfalz -0.77836 0.09538 -8.160 0.000000000000000334  
## StateSaarland -1.13274 0.09552 -11.859 < 0.0000000000000002  
## StateSachsen -1.08280 0.09541 -11.349 < 0.0000000000000002  
## StateSachsen-Anhalt -1.36482 0.09544 -14.300 < 0.0000000000000002  
## StateSchleswig-Holstein -0.36652 0.09541 -3.842 0.000122  
## StateThüringen -1.20943 0.09555 -12.657 < 0.0000000000000002  
##   
## (Intercept) \*\*\*  
## StateBaden-Württemberg \*\*   
## StateBayern   
## StateBerlin \*   
## StateBrandenburg \*\*\*  
## StateBremen \*\*\*  
## StateHamburg \*   
## StateHessen \*\*\*  
## StateMecklenburg-Vorpommern \*\*\*  
## StateNiedersachsen \*\*\*  
## StateNordrhein-Westfalen \*\*\*  
## StateRheinland-Pfalz \*\*\*  
## StateSaarland \*\*\*  
## StateSachsen \*\*\*  
## StateSachsen-Anhalt \*\*\*  
## StateSchleswig-Holstein \*\*\*  
## StateThüringen \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1039137 on 10317 degrees of freedom  
## Residual deviance: 812246 on 10301 degrees of freedom  
## AIC: 884878  
##   
## Number of Fisher Scoring iterations: 5

exp(coef(glm.sq.price))

## (Intercept) StateBaden-Württemberg   
## 4227.2727273 0.7635056   
## StateBayern StateBerlin   
## 0.8919084 1.2474312   
## StateBrandenburg StateBremen   
## 0.7105675 0.6374919   
## StateHamburg StateHessen   
## 1.2335789 0.7007400   
## StateMecklenburg-Vorpommern StateNiedersachsen   
## 0.4616627 0.4559654   
## StateNordrhein-Westfalen StateRheinland-Pfalz   
## 0.5942937 0.4591593   
## StateSaarland StateSachsen   
## 0.3221483 0.3386464   
## StateSachsen-Anhalt StateSchleswig-Holstein   
## 0.2554273 0.6931439   
## StateThüringen   
## 0.2983668

ggplot(data = data,  
 mapping = aes(y = Price/Living\_space,  
 x = State)) +   
 geom\_point() +  
 geom\_smooth(method = "glm",   
 se = FALSE,  
 method.args = list(family = "binomial"))

## `geom\_smooth()` using formula 'y ~ x'

 If we compare the Residual deviance and the corresponding degrees of freedom in the summary output we would expect in an truly Poisson distributed data hat the residual deviance and the degrees of freedom would be approximately the same value. Therefore it could be overdispersed here and we use the “quasibinomial” family.

glm.sq.price <- glm(cbind(Price, Living\_space)~ State,  
 family = "quasibinomial",  
 data = data)  
  
summary(glm.sq.price)

##   
## Call:  
## glm(formula = cbind(Price, Living\_space) ~ State, family = "quasibinomial",   
## data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -59.255 -6.366 -1.195 3.473 61.109   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.3493 0.9998 8.351 <0.0000000000000002 \*\*\*  
## StateBaden-Württemberg -0.2698 1.0000 -0.270 0.787   
## StateBayern -0.1144 1.0000 -0.114 0.909   
## StateBerlin 0.2211 1.0012 0.221 0.825   
## StateBrandenburg -0.3417 1.0004 -0.342 0.733   
## StateBremen -0.4502 1.0066 -0.447 0.655   
## StateHamburg 0.2099 1.0072 0.208 0.835   
## StateHessen -0.3556 1.0000 -0.356 0.722   
## StateMecklenburg-Vorpommern -0.7729 1.0006 -0.772 0.440   
## StateNiedersachsen -0.7853 1.0000 -0.785 0.432   
## StateNordrhein-Westfalen -0.5204 0.9999 -0.520 0.603   
## StateRheinland-Pfalz -0.7784 1.0000 -0.778 0.436   
## StateSaarland -1.1327 1.0015 -1.131 0.258   
## StateSachsen -1.0828 1.0003 -1.083 0.279   
## StateSachsen-Anhalt -1.3648 1.0006 -1.364 0.173   
## StateSchleswig-Holstein -0.3665 1.0003 -0.366 0.714   
## StateThüringen -1.2094 1.0018 -1.207 0.227   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for quasibinomial family taken to be 109.9222)  
##   
## Null deviance: 1039137 on 10317 degrees of freedom  
## Residual deviance: 812246 on 10301 degrees of freedom  
## AIC: NA  
##   
## Number of Fisher Scoring iterations: 5

exp(coef(glm.sq.price))

## (Intercept) StateBaden-Württemberg   
## 4227.2727273 0.7635056   
## StateBayern StateBerlin   
## 0.8919084 1.2474312   
## StateBrandenburg StateBremen   
## 0.7105675 0.6374919   
## StateHamburg StateHessen   
## 1.2335789 0.7007400   
## StateMecklenburg-Vorpommern StateNiedersachsen   
## 0.4616627 0.4559654   
## StateNordrhein-Westfalen StateRheinland-Pfalz   
## 0.5942937 0.4591593   
## StateSaarland StateSachsen   
## 0.3221483 0.3386464   
## StateSachsen-Anhalt StateSchleswig-Holstein   
## 0.2554273 0.6931439   
## StateThüringen   
## 0.2983668

The dispersion parameter ist now 109.92, This implies the variance increases faster than linearly. Anyway in this case there is no evidence that the State have an impact on the response variable.

#install.packages('mltools')  
library(mltools)  
  
# Resulting bins have an equal number of observations in each group  
data[, "wt2"] <- bin\_data(data$Price, bins=4, binType = "quantile")  
  
# Resulting bins are equally spaced from min to max  
data[, "wt3"] <- bin\_data(data$Price, bins=4, binType = "explicit")  
  
# Or if you'd rather define the bins yourself  
data[, "wt4"] <- bin\_data(data$Price, bins=c(-Inf, 250, 322, Inf), binType = "explicit")  
head(data)

## Price Type Living\_space Lot Rooms Bedrooms Bathrooms Floors  
## 1 498000 Multiple dwelling 106.00 229 5 3 1 2  
## 2 495000 Mid-terrace house 140.93 517 6 3 2 NA  
## 3 749000 Farmhouse 162.89 82 5 3 2 4  
## 4 259000 Farmhouse 140.00 814 4 NA 2 2  
## 5 469000 Multiple dwelling 115.00 244 4 2 1 NA  
## 6 1400000 Mid-terrace house 310.00 860 8 NA NA 3  
## Year\_built Furnishing\_quality Year\_renovated Condition Heating  
## 1 2005 normal NA refurbished central heating  
## 2 1994 basic NA refurbished stove heating  
## 3 2013 NA dilapidated stove heating  
## 4 1900 basic 2000 fixer-upper central heating  
## 5 1968 refined 2019 refurbished central heating  
## 6 1969 basic NA maintained   
## Energy\_source Energy\_efficiency\_class State City  
## 1 natural gas D Baden-Württemberg Bodenseekreis  
## 2 Baden-Württemberg Konstanz (Kreis)  
## 3 district heating B Baden-Württemberg Esslingen (Kreis)  
## 4 electricity G Baden-Württemberg Waldshut (Kreis)  
## 5 oil F Baden-Württemberg Esslingen (Kreis)  
## 6 oil Baden-Württemberg Stuttgart  
## Place Garages Garagetype wt2  
## 1 Bermatingen 2 Parking lot [399000, 649900)  
## 2 Engen 7 Parking lot [399000, 649900)  
## 3 Ostfildern 1 Garage [649900, 13000000]  
## 4 Bonndorf im Schwarzwald 1 Garage [250000, 399000)  
## 5 Leinfelden-Echterdingen 1 Garage [399000, 649900)  
## 6 Süd 2 Garage [649900, 13000000]  
## wt3 wt4  
## 1 [10000, 3257500) [322, Inf]  
## 2 [10000, 3257500) [322, Inf]  
## 3 [10000, 3257500) [322, Inf]  
## 4 [10000, 3257500) [322, Inf]  
## 5 [10000, 3257500) [322, Inf]  
## 6 [10000, 3257500) [322, Inf]

glm.roomswt2 <- glm(Rooms ~ wt2,  
family = "poisson",  
data = data)  
  
summary(glm.roomswt2)

##   
## Call:  
## glm(formula = Rooms ~ wt2, family = "poisson", data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2229 -0.8439 -0.3171 0.4433 4.5174   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.922092 0.003783 508.099 < 0.0000000000000002 \*\*\*  
## wt2.L 0.235488 0.007507 31.369 < 0.0000000000000002 \*\*\*  
## wt2.Q 0.056346 0.007566 7.447 0.0000000000000952 \*\*\*  
## wt2.C 0.041856 0.007624 5.490 0.0000000401941232 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 13951 on 10317 degrees of freedom  
## Residual deviance: 12833 on 10314 degrees of freedom  
## AIC: 51023  
##   
## Number of Fisher Scoring iterations: 4

set.seed(99)  
sim.data.rooms.Poissonwt2 <- simulate(glm.roomswt2)  
##  
NROW(sim.data.rooms.Poissonwt2)

## [1] 10318

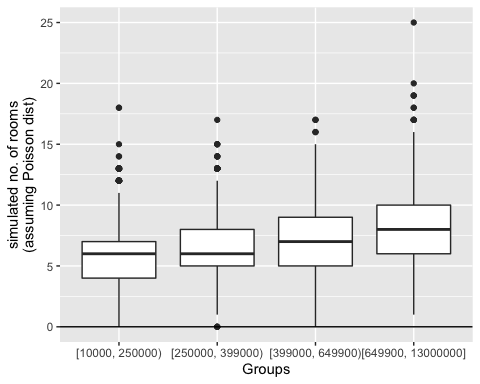
head(sim.data.rooms.Poissonwt2)

## sim\_1  
## 1 7  
## 2 4  
## 3 10  
## 4 13  
## 5 7  
## 6 14

tail(sim.data.rooms.Poissonwt2)

## sim\_1  
## 10313 10  
## 10314 16  
## 10315 11  
## 10316 5  
## 10317 5  
## 10318 9

library(ggplot2)  
ggplot(mapping = aes(y = sim.data.rooms.Poissonwt2$sim\_1,  
x = data$wt2)) +  
geom\_boxplot() +  
geom\_hline(yintercept = 0) +  
ylab("simulated no. of rooms\n(assuming Poisson dist)") +  
xlab("Groups")

 **GLM - Binary Model**

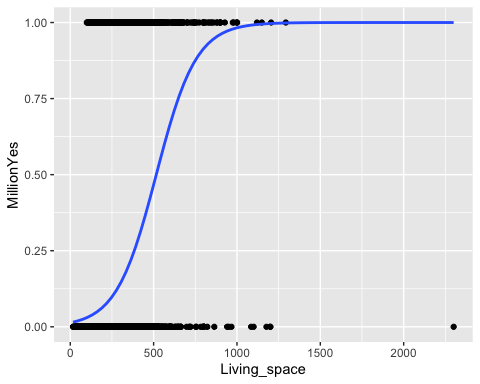
Let’s fit a binary model

data$MillionYes <- ifelse(data$Price > 1000000, 1, 0)  
data$MillionYes

Let’s fit a logistic regression model and add fit to the this graph

glm.binary <- glm(MillionYes ~ Living\_space,  
 family = "binomial",  
 data = data)  
  
ggplot(data = glm.binary,  
 mapping = aes(y = MillionYes,  
 x = Living\_space)) +   
 geom\_point() +  
 geom\_smooth(method = "glm",   
 se = FALSE,  
 method.args = list(family = "binomial"))

## `geom\_smooth()` using formula 'y ~ x'



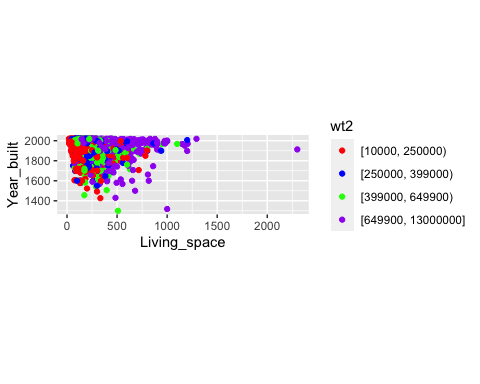
# Week 4 - Support Vector Machines

data$wt5 <- factor(ifelse(data$Price > 600000, 1, -1))  
data$wt5 = as.factor(data$wt5)  
str(data)

## 'data.frame': 10318 obs. of 25 variables:  
## $ Price : num 498000 495000 749000 259000 469000 1400000 3500000 630000 364000 1750000 ...  
## $ Type : chr "Multiple dwelling" "Mid-terrace house" "Farmhouse" "Farmhouse" ...  
## $ Living\_space : num 106 141 163 140 115 ...  
## $ Lot : num 229 517 82 814 244 860 5300 406 973 1460 ...  
## $ Rooms : int 5 6 5 4 4 8 13 10 10 6 ...  
## $ Bedrooms : num 3 3 3 NA 2 NA NA NA 4 4 ...  
## $ Bathrooms : num 1 2 2 2 1 NA 4 NA 4 2 ...  
## $ Floors : num 2 NA 4 2 NA 3 NA 3 2 3 ...  
## $ Year\_built : num 2005 1994 2013 1900 1968 ...  
## $ Furnishing\_quality : chr "normal" "basic" "" "basic" ...  
## $ Year\_renovated : num NA NA NA 2000 2019 ...  
## $ Condition : chr "refurbished" "refurbished" "dilapidated" "fixer-upper" ...  
## $ Heating : chr "central heating" "stove heating" "stove heating" "central heating" ...  
## $ Energy\_source : chr "natural gas" "" "district heating" "electricity" ...  
## $ Energy\_efficiency\_class: chr "D" "" "B" "G" ...  
## $ State : chr "Baden-Württemberg" "Baden-Württemberg" "Baden-Württemberg" "Baden-Württemberg" ...  
## $ City : chr "Bodenseekreis" "Konstanz (Kreis)" "Esslingen (Kreis)" "Waldshut (Kreis)" ...  
## $ Place : chr "Bermatingen" "Engen" "Ostfildern" "Bonndorf im Schwarzwald" ...  
## $ Garages : num 2 7 1 1 1 2 7 2 8 2 ...  
## $ Garagetype : chr "Parking lot" "Parking lot" "Garage" "Garage" ...  
## $ wt2 : Ord.factor w/ 4 levels "[10000, 250000)"<..: 3 3 4 2 3 4 4 3 2 4 ...  
## $ wt3 : Ord.factor w/ 4 levels "[10000, 3257500)"<..: 1 1 1 1 1 1 2 1 1 1 ...  
## $ wt4 : Ord.factor w/ 3 levels "[-Inf, 250)"<..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ MillionYes : num 0 0 0 0 0 1 1 0 0 1 ...  
## $ wt5 : Factor w/ 2 levels "-1","1": 1 1 2 1 1 2 2 2 1 2 ...

# Load ggplot2  
library(ggplot2)  
# Plot x2 vs. x1, colored by y  
scatter\_plot<- ggplot(data = data, aes(x = Living\_space, y = Year\_built, color = wt2  
   
 )) +   
 # Add a point layer  
 geom\_point() +   
 scale\_color\_manual(values = c("red", "blue","green", "purple")) +  
 # Specify equal coordinates  
 coord\_equal()  
   
scatter\_plot

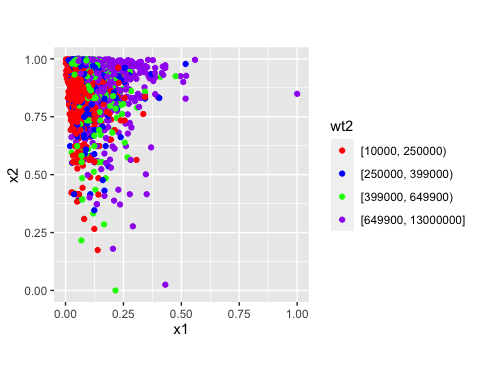
## Warning: Removed 666 rows containing missing values (geom\_point).



x1 <- scales::rescale(data$Living\_space, to=c(0,1))  
x2 <- scales::rescale(data$Year\_built, to=c(0,1))  
x3 <- scales::rescale(data$Rooms, to=c(0,1))  
x4 <- scales::rescale(data$Lot, to=c(0,1))

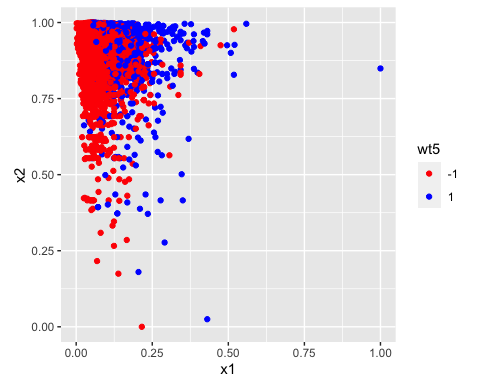
# Load ggplot2  
library(ggplot2)  
# Plot x2 vs. x1, colored by y  
scatter\_plot<- ggplot(data = data, aes(x = x1, y = x2, color = wt2  
   
 )) +   
 # Add a point layer  
 geom\_point() +   
 scale\_color\_manual(values = c("red", "blue","green", "purple")) +  
 # Specify equal coordinates  
 coord\_equal()  
   
scatter\_plot

## Warning: Removed 666 rows containing missing values (geom\_point).

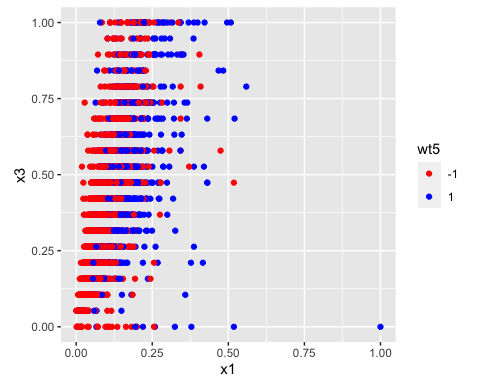


# Load ggplot2  
library(ggplot2)  
# Plot x2 vs. x1, colored by y  
scatter\_plot<- ggplot(data = data, aes(x = x1, y = x2, color = wt5  
   
 )) +   
 # Add a point layer  
 geom\_point() +   
 scale\_color\_manual(values = c("red", "blue")) +  
 # Specify equal coordinates  
 coord\_equal()  
   
scatter\_plot

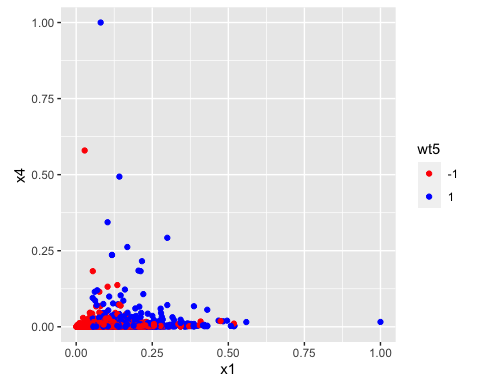
## Warning: Removed 666 rows containing missing values (geom\_point).



# Load ggplot2  
library(ggplot2)  
# Plot x2 vs. x1, colored by y  
scatter\_plot<- ggplot(data = data, aes(x = x1, y = x3, color = wt5  
   
 )) +   
 # Add a point layer  
 geom\_point() +   
 scale\_color\_manual(values = c("red", "blue")) +  
 # Specify equal coordinates  
 coord\_equal()  
   
scatter\_plot



# Load ggplot2  
library(ggplot2)  
# Plot x2 vs. x1, colored by y  
scatter\_plot<- ggplot(data = data, aes(x = x1, y = x4, color = wt5  
   
 )) +   
 # Add a point layer  
 geom\_point() +   
 scale\_color\_manual(values = c("red", "blue")) +  
 # Specify equal coordinates  
 coord\_equal()  
   
scatter\_plot



testdf <- data.frame(x2,x1,data$wt5)

#install.packages("e1071")  
library(e1071)

##   
## Attaching package: 'e1071'

## The following object is masked from 'package:mltools':  
##   
## skewness

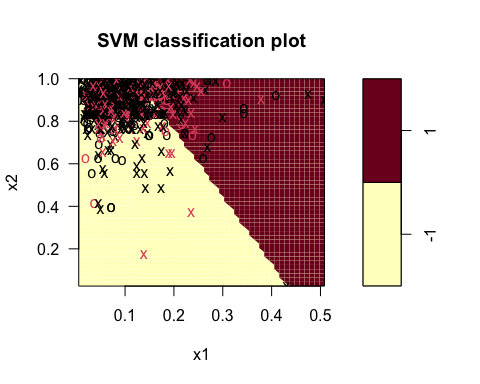
# Print average accuracy and standard deviation  
accuracy <- rep(NA, 100)  
set.seed(2)  
# Calculate accuracies for 100 training/test partitions  
for (i in 1:100){  
 testdf[, "train"] <- ifelse(runif(nrow(testdf)) < 0.8, 1, 0)  
 trainset <- testdf[testdf$train == 1, ]  
 testset <- testdf[testdf$train == 0, ]  
 trainColNum <- grep("train", names(trainset))  
 trainset <- trainset[, -trainColNum]  
 testset <- testset[, -trainColNum]  
 svm\_model <- svm(data.wt5 ~ ., data = trainset, type = "C-classification", kernel = "linear")  
 pred\_test <- predict(svm\_model, testset)  
 accuracy[i] <- mean(pred\_test == testset$data.wt5)  
}  
# Print average accuracy and standard deviation  
mean(accuracy)

## [1] 0.6886003

sd(accuracy)

## [1] 0.009164209

plot(svm\_model,testset)



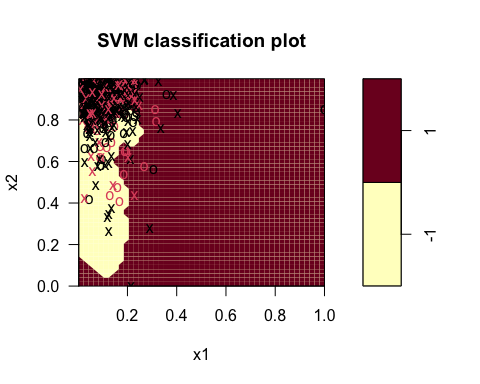
# Print average accuracy and standard deviation  
accuracy <- rep(NA, 10)  
set.seed(2)  
# Calculate accuracies for 100 training/test partitions  
for (i in 1:10){  
 testdf[, "train"] <- ifelse(runif(nrow(testdf)) < 0.8, 1, 0)  
 trainset <- testdf[testdf$train == 1, ]  
 testset <- testdf[testdf$train == 0, ]  
 trainColNum <- grep("train", names(trainset))  
 trainset <- trainset[, -trainColNum]  
 testset <- testset[, -trainColNum]  
 svm\_model <- svm(data.wt5 ~ ., data = trainset, type = "C-classification", kernel = "radial")  
 pred\_test <- predict(svm\_model, testset)  
 accuracy[i] <- mean(pred\_test == testset$data.wt5)  
}  
  
# Print average accuracy and standard deviation  
mean(accuracy)

## [1] 0.6656181

sd(accuracy)

## [1] 0.007764355

plot(svm\_model,testset)



# Week 5 - Neural Networks

## ANN - neuralnet package

preparing data for neuralnet

library(tidyverse)  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

## The following object is masked from 'package:purrr':  
##   
## transpose

library(neuralnet)

##   
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':  
##   
## compute

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

## The following object is masked from 'package:survival':  
##   
## cluster

#str(data)  
#apply(data,2,function(x) sum(is.na(x)))  
  
df <- data.frame(data$Price, data$Living\_space, data$Lot, data$Rooms, data$Year\_built, data$Garages)  
  
df.house <- na.omit(df)  
  
#dummy <- dummyVars(" ~ .", data=df.house)  
#df.house <- data.frame(predict(dummy, newdata = df.house))   
any(is.na(df.house))

## [1] FALSE

str(df.house)

## 'data.frame': 7974 obs. of 6 variables:  
## $ data.Price : num 498000 495000 749000 259000 469000 1400000 3500000 630000 364000 1750000 ...  
## $ data.Living\_space: num 106 141 163 140 115 ...  
## $ data.Lot : num 229 517 82 814 244 860 5300 406 973 1460 ...  
## $ data.Rooms : int 5 6 5 4 4 8 13 10 10 6 ...  
## $ data.Year\_built : num 2005 1994 2013 1900 1968 ...  
## $ data.Garages : num 2 7 1 1 1 2 7 2 8 2 ...  
## - attr(\*, "na.action")= 'omit' Named int [1:2344] 25 26 29 38 44 50 57 70 72 75 ...  
## ..- attr(\*, "names")= chr [1:2344] "25" "26" "29" "38" ...

mean(data$Price)

## [1] 540586.6

**Prepare for Training**

**Fit the Network**

And calculate the RMSE

## ANN - caret package

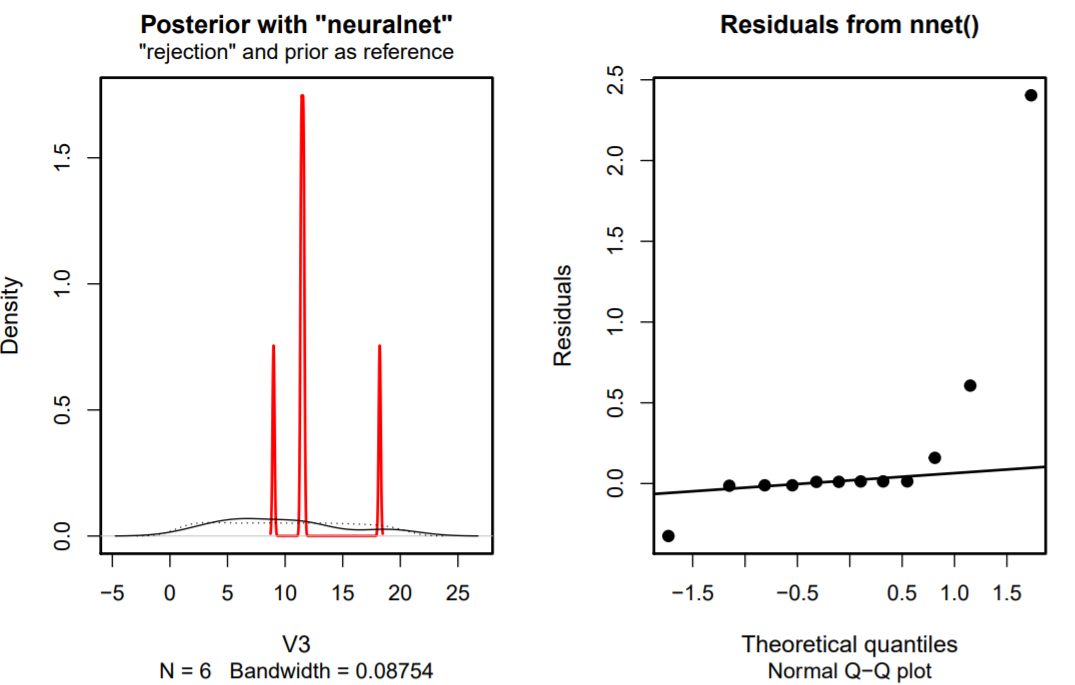
#str(df.house)  
  
set.seed(42)  
tuGrid <- expand.grid(.layer1=c(1:4), .layer2=c(0,2), .layer3=c(0))  
  
trCtrl <- trainControl(  
 method = 'repeatedcv',   
 number = 5,   
 repeats = 5,   
 returnResamp = 'final'  
)  
  
models <- train(  
 x = df.house %>% select(-data.Price),  
 y = df.house\_scaled %>% pull(data.Price),  
 method = 'neuralnet', metric = 'RMSE',   
 linear.output = TRUE,  
 #be careful, does only work on x!  
 preProcess = c('center', 'scale'),  
 tuneGrid = tuGrid,  
 trControl = trCtrl  
)

plot(models)

plot(models$finalModel)

# Week 6 - Agent-based Modelling and Approximate Bayesian Computation

## Result



ABM