MPM.02 Groupwork

Felix, Regis, Sofie, Philipp

06/01/2022

Contents

1 Introduction **3**

2 Graphical Data Exploration **3**
2.1 Influnece of Weather upon Bike Rentals . . . 3
2.1.1 Exploration of Continuous Variables . . . 3
2.1.2 Exploration of Categorical Variables . . . 6
2.2 Influnece of Timely and Seasonal aspects upon Bike Rentals . . . 6
2.2.1 Exploration of Categorical Variables . . . 6
2.2.1.1 Seasonal Influence . . . 6
2.2.1.2 Timely Influence . . . 7

3 Linear Regression Model **8**
3.1 Data Preperation for the Linear Model . . . 8
3.1.1 Set Categorical Variables as Factors . . . 8
3.1.2 Re-Transform the Continious Weather Variables . . . 8
3.1.3 Split data into train and test dataset . . . 9
3.2 Multiple Linear Regression Model with Continious Weather Predictors . . . 9
3.2.1 Simplification of Model lm.fit.0.train . . . 9
3.2.2 Accessing the model accuracy . . . 10
3.2.2.1 Compare R-squared Values . . . 10
3.2.2.2 Compare RMSE Values . . . 10
3.3 Multiple Linear Regression Model with Continious and Categorical Variables . . . 11
3.3.1 Simplification of Model lm.fit.2.train . . . 11
3.3.2 Accessing the model accuracy . . . 11
3.3.2.1 Compare R-Squared Values . . . 11
3.3.2.2 Compare RMSE Values . . . 11
3.4 Summary . . . 13
3.4.1 Is there a relationship betweeen the independent variables and the response variable? 13
3.4.2 How strong is the relationship and how accurate is the model? . . . 13
3.4.3 How large is the effect of each predctor on cnt? . . . 13
3.4.4 Are there Potential Problems of Fitting this data with a Linear Regression Model . . 14

4 General Additive Model (GAM) **16**
4.1 Check if model with non-linear function ist better . . . 16
4.1.1 Compare RMSE Values . . . 17
4.2 Add categorical variables to GAM Model gam_04 . . . 17
4.3 Summary . . . 17

5 GLM Binomial and Poisson **18**
5.1 Plotting and inspecting the data . . . 18
5.2 Creating training and testing sets . . . 19

5.3 Creating a binomial model . . . 19
5.3.1 Fitting the binomial model with “train” data . . . 20
5.3.2 Conclusion: GLM Binomial . . . 24
5.4 Creating a Poisson Model . . . 24
5.4.1 Fitting the poisson model with “train” data . . . 25
5.4.2 Conclusion: GLM Poisson . . . 27

6 Support Vector Machine **27**
6.1 Load the data . . . 27
6.2 Prepare the data . . . 27
6.3 Factor ordinal variables . . . 27
6.4 Plot data . . . 28
6.4.1 Rentals by Season and Temperature . . . 28
6.4.2 Rentals by Season, Temperature and Year . . . 29
6.4.3 Rental and Temperature by Seasion . . . 30
6.4.4 Rentals and Temperature by Weather . . . 31
6.5 Train - test split . . . 33
6.6 SVM model . . . 33
6.6.1 Kfold cross validation . . . 33
6.6.2 Linear kernel . . . 33
6.6.2.1 Save resp. Load Model . . . 34
6.6.2.2 Save resp. Load Model . . . 34
6.6.3 Prediction with linear kernel . . . 34
6.6.4 Polynomial kernel . . . 34
6.6.4.1 Save resp. Load Model . . . 35
6.6.5 Prediction with polynomaial kernel . . . 35
6.6.6 Radial kernel . . . 35
6.6.6.1 Save resp. Load Model . . . 35
6.6.7 Prediction with radial kernel . . . 36
6.6.8 Model comparision . . . 36

7 Artificial Neural Network **36**
7.1 Data Preparation . . . 36
7.1.1 Load data . . . 36
7.1.2 Break multi factor variables down into single factor variables . . . 36
7.1.3 Normalize dependent variable . . . 36
7.2 ANN Model 1: Computing a RAnge of Model . . . 37
7.2.1 Split Data into Train and Test Partition . . . 37
7.2.2 Set up Parameter Range . . . 37
7.2.3 Train Models . . . 37
7.2.4 Save resp. Load Model . . . 37
7.2.5 Plot Models . . . 37
7.2.6 Compute Prediction with best Model . . . 38
7.2.7 Root Mean Square . . . 38
7.2.8 Plot Prediction against Real Data . . . 38
7.3 ANN Model 2: Layer 4/0/0, Threshold 0.01 . . . 39
7.3.1 Split Data into Train and Test Partition . . . 39
7.3.2 Train Model . . . 39
7.3.3 Save resp. Load Model . . . 39
7.3.4 Compute Prediction . . . 39
7.3.5 Root Mean Square . . . 40
7.3.6 Plot Prediction against Real Data . . . 40
7.4 ANN Model 4: more Predictors, Layer 4/0/0, Threshold 0.05 . . . 40
7.4.1 Train Model . . . 40

7.4.2 Load model (computed in advance) . . . 40
7.4.3 Compute Prediction . . . 41
7.4.4 Root Mean Square . . . 41
7.4.5 Plot Model Performance . . . 41
7.5 ANN Models 5: Computing a Range of Model . . . 41
7.5.1 Set up Parameter Range . . . 41
7.5.2 Train Models . . . 41
7.5.3 Plot Models . . . 42
7.5.4 Compute Prediction . . . 42
7.5.5 Root Mean Square . . . 42
7.5.6 Plot Prediction against Real Data . . . 42
7.5.7 Compute Prediction . . . 43
7.5.8 Root Mean Square . . . 43
7.5.9 Plot Prediction against Real Data . . . 43
7.6 ANN Model 9: New Pred.: Year, Layer 7/3/0, Threshold 0.01 . . . 44
7.6.1 Compute Prediction . . . 44
7.6.2 Root Mean Square . . . 44
7.6.3 Plot Prediction against Real Data . . . 44
7.7 ANN Models 10: Computing a RAnge of Model . . . 45
7.7.1 Set up Parameter Range . . . 45
7.7.2 Train Models . . . 45
7.8 ANN Model 11 . . . 46
7.8.1 Compute Prediction . . . 46
7.8.2 Root Mean Square . . . 46
7.8.3 Plot Prediction against Real Data . . . 46
7.9 Further models . . . 47
7.10 ANN Reflection & Conclsion: . . . 47

8 Optimization Problem **47**
8.1 Which model should the company use? . . . 48

1 Introduction

In this project work for machine learning 1, we would like to examine a dataset of bicycle rental numbers and build different models to predict the number of rentals at a given time.

Obviously, the number of rentals depends on the weather as well as on temporal or seasonal factors. These factors will be investigated.

The historical data is from the Capital BikeShare system for Washington D.C. USA.The data is available for the year 2011 to 2012.The dataset also contains weather data at the given time points. To form a usecase here, the company Capital BikeShare would like to predict the amount of rentals at a given time to be able to better plan the work resources in the future. We as a data science startup were engaged to estimate the number of bikes based on the data.

2 Graphical Data Exploration

2.1 Influnece of Weather upon Bike Rentals

In this chapter we would like to explore how different weather conditions influence the bike rental numbers.

2.1.1 Exploration of Continuous Variables

We want to explore how different weather situation influences the number of bike rentals. For that we have following 4 continuous weather variables at hand which we are going to examine:

- temp: Normalized temperature in Celsius. t_min [-8], t_max [+39]
- atemp: Normalized feels like temperature in Celsius. t_min [-16], t_max [+50]
- hum: Normalized humidity. The values are divided with max. of 100
- windspeed: Normalized wind speed. The values are divided with min. of 67

The continuous weather variables are normalized as follows in the dataset:

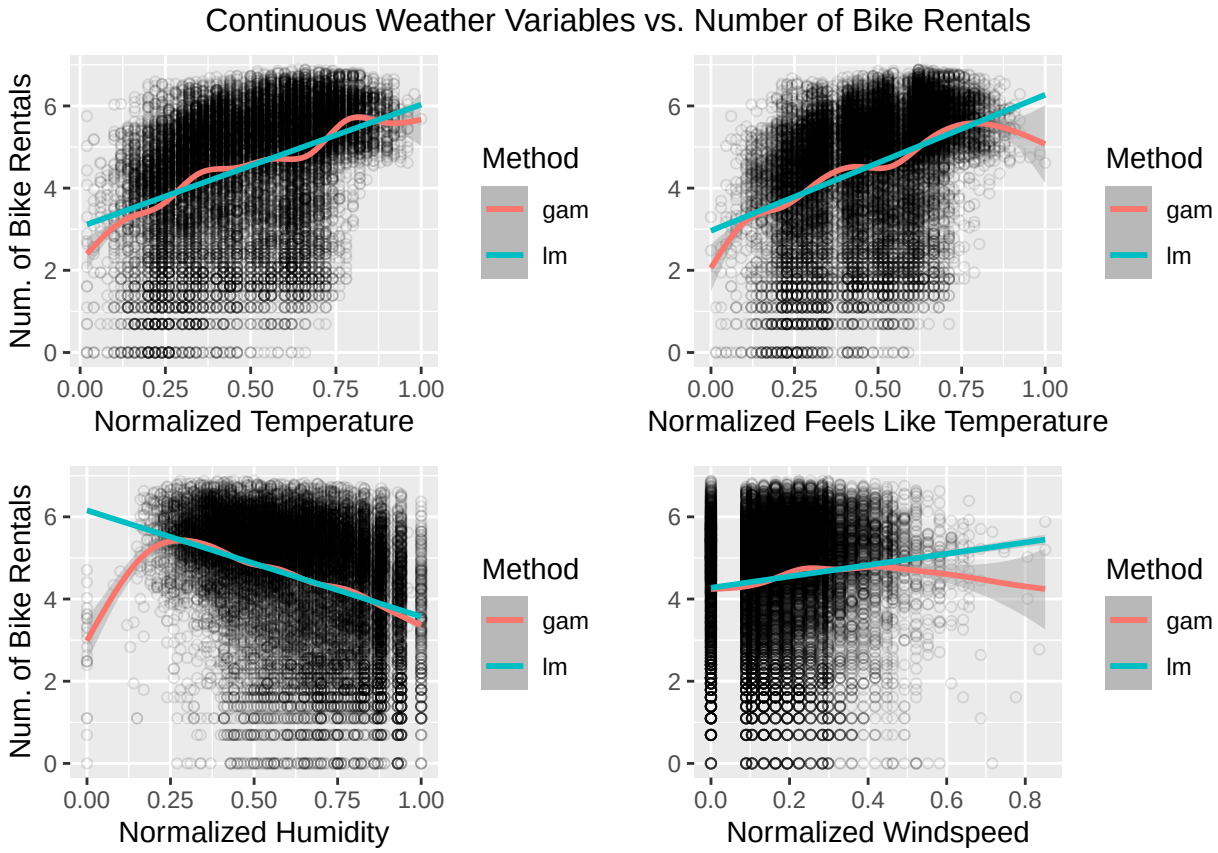
```
## 'data.frame':    17379 obs. of  17 variables:
##  $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ dteday    : chr  "2011-01-01" "2011-01-01" "2011-01-01" "2011-01-01" ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hr        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday   : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weathersit: int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  0.24 0.22 0.22 0.24 0.24 0.24 0.22 0.2 0.24 0.32 ...
##  $ atemp     : num  0.288 0.273 0.273 0.288 0.288 ...
##  $ hum       : num  0.81 0.8 0.8 0.75 0.75 0.75 0.8 0.86 0.75 0.76 ...
##  $ windspeed : num  0 0 0 0 0 0.0896 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ cnt       : int  16 40 32 13 1 1 2 3 8 14 ...
```

```
##     instant         dteday              season           yr        
##  Min.   :    1   Length:17379       Min.   :1.000   Min.   :0.0000  
##  1st Qu.: 4346   Class :character   1st Qu.:2.000   1st Qu.:0.0000  
##  Median : 8690   Mode  :character   Median :3.000   Median :1.0000  
##  Mean   : 8690                      Mean   :2.502   Mean   :0.5026  
##  3rd Qu.:13034                      3rd Qu.:3.000   3rd Qu.:1.0000  
##  Max.   :17379                      Max.   :4.000   Max.   :1.0000  
##       mnth              hr           holiday           weekday      
##  Min.   : 1.000   Min.   : 0.00   Min.   :0.00000   Min.   :0.000  
##  1st Qu.: 4.000   1st Qu.: 6.00   1st Qu.:0.00000   1st Qu.:1.000  
##  Median : 7.000   Median :12.00   Median :0.00000   Median :3.000  
##  Mean   : 6.538   Mean   :11.55   Mean   :0.02877   Mean   :3.004  
##  3rd Qu.:10.000   3rd Qu.:18.00   3rd Qu.:0.00000   3rd Qu.:5.000  
##  Max.   :12.000   Max.   :23.00   Max.   :1.00000   Max.   :6.000  
##    workingday       weathersit         temp           atemp      
##  Min.   :0.0000   Min.   :1.000   Min.   :0.020   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.340   1st Qu.:0.3333  
##  Median :1.0000   Median :1.000   Median :0.500   Median :0.4848  
##  Mean   :0.6827   Mean   :1.425   Mean   :0.497   Mean   :0.4758  
##  3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:0.660   3rd Qu.:0.6212  
##  Max.   :1.0000   Max.   :4.000   Max.   :1.000   Max.   :1.0000  
##       hum           windspeed          casual         registered    
##  Min.   :0.0000   Min.   :0.0000   Min.   :  0.00   Min.   :  0.0  
##  1st Qu.:0.4800   1st Qu.:0.1045   1st Qu.:  4.00   1st Qu.: 34.0  
##  Median :0.6300   Median :0.1940   Median : 17.00   Median :115.0  
##  Mean   :0.6272   Mean   :0.1901   Mean   : 35.68   Mean   :153.8  
##  3rd Qu.:0.7800   3rd Qu.:0.2537   3rd Qu.: 48.00   3rd Qu.:220.0  
##  Max.   :1.0000   Max.   :0.8507   Max.   :367.00   Max.   :886.0  
##       cnt           
```

```
##  Min.   :  1.0
##  1st Qu.: 40.0
##  Median :142.0
##  Mean   :189.5
##  3rd Qu.:281.0
##  Max.   :977.0
```

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

For some models it may be useful to use the normalized variables and for other we may re-transform the variables. If so, we will state that in the corresponding chapter.



We consider the response variables as an amount altough we are a bit unsure here. However the results in the next chapter shows that using the log function improves the linear regression model quite a bit.

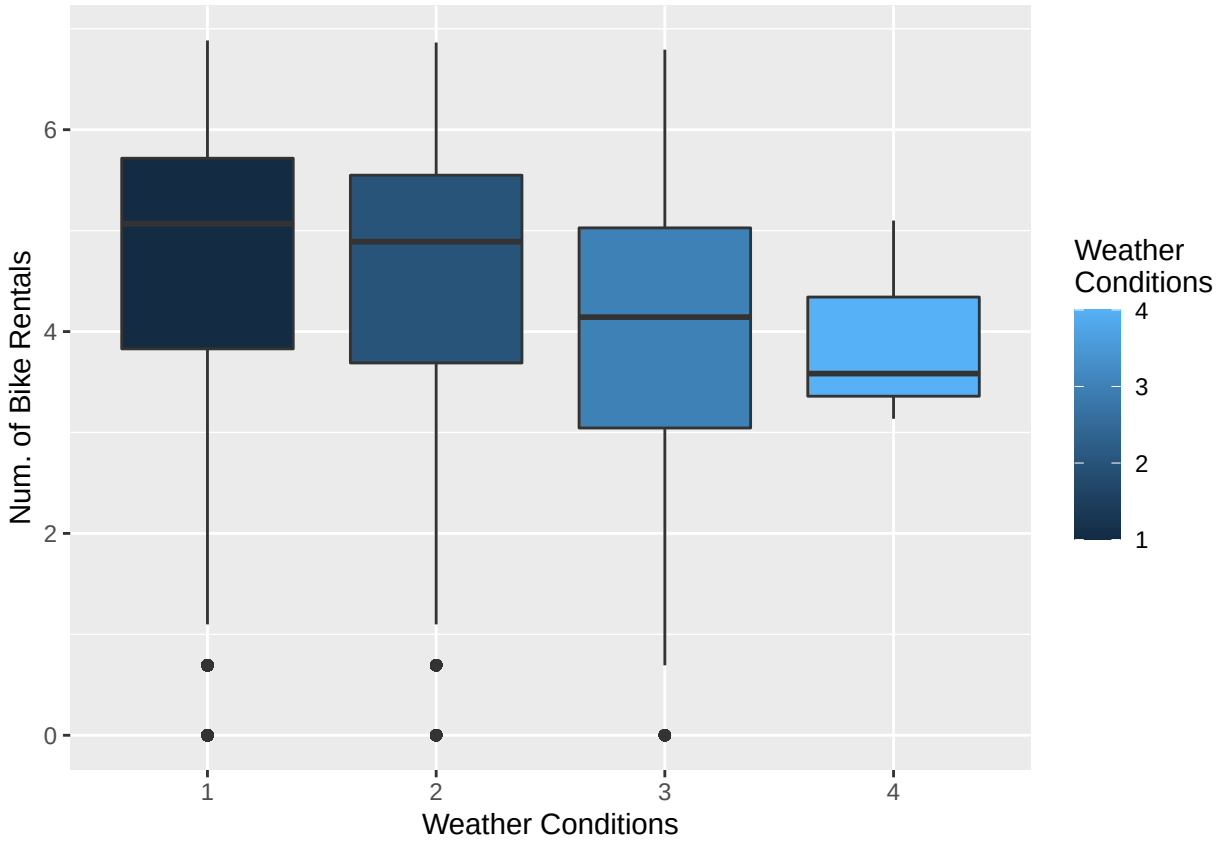
In all 4 scatter plots above the linear model lines (green) as well as the gam model lines (red) are plotted. For both temperature variables the rentals seems to increase until the temperature reaches a certain value.At the beginning of the observations (low temp.) ans well as the end (high temp.) the linear model seems to overestimate the number of rentals. For the normalized humidity there is a decrease in rentals with increasing humidity. The is not the case between 0 and 25 % humidity where the linear model overestimates the number of rentals. We assume that in very cold and very hot weather the humidity is rather low and so the number of bike rentals because of the influence of temperature. In conclusion of the above exploration, it can be stated that the behavior cannot be completely represented with a linear model. There are nonlinear aspects which should be considered in the prediction models. In the next chapter we would like to examine the categorical variables which are available in the dataset.

2.1.2 Exploration of Categorical Variables

The data set contains 4 different weather categories which are described as follows:

- Category 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- Category 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- Category 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- Category 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Since we are considering the response variables count data we log transform the variable with the natural logarithm in all of the following plots.



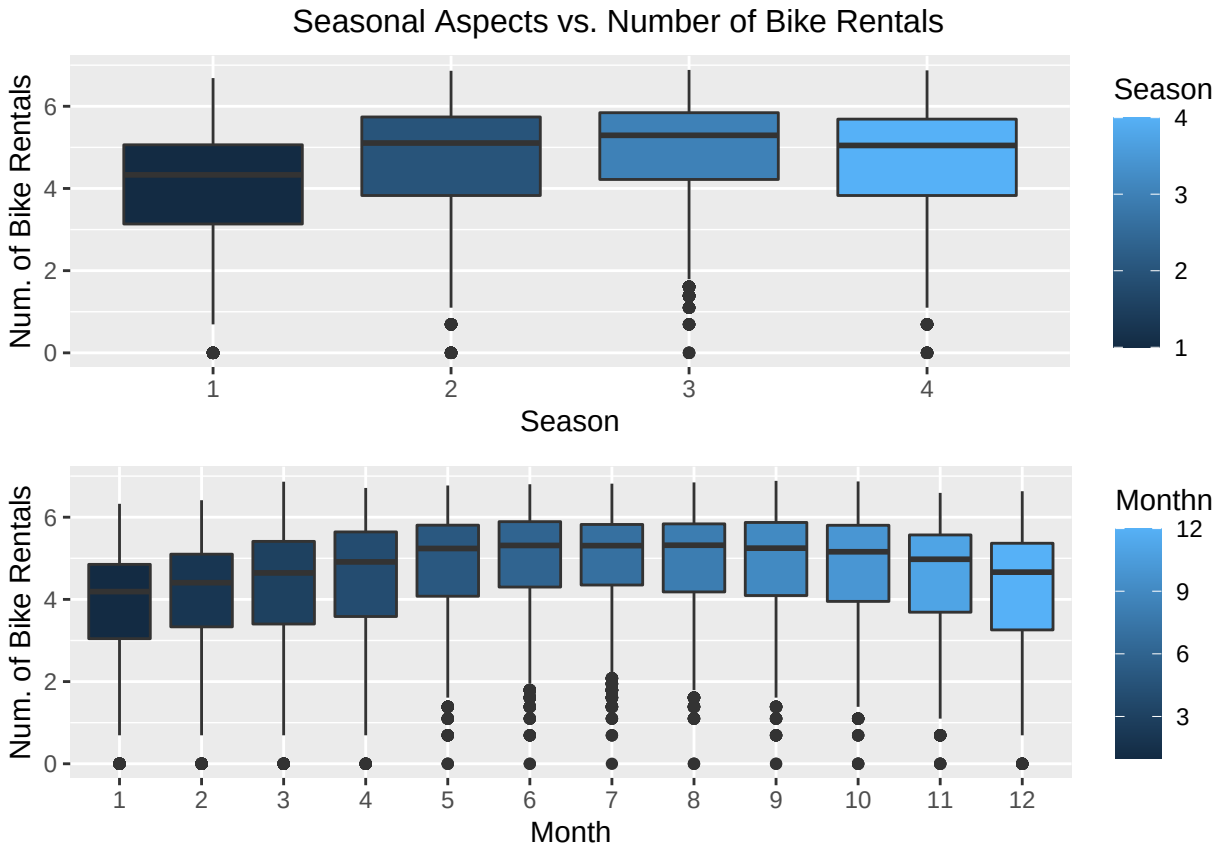
The boxplot above indicates, that in situation 1 and 2 the most bikes are rented whereas there is seems to be a drop visible for condition 3 and the drop is even more significant for situation 4 where severe weather conditions are present.This rental behavior is as we have expected since few people ride bicycles in bad weather situations. In the next chapter we are inspecting the seasonal and timely influence upon the rental situation.

2.2 Influnece of Timely and Seasonal aspects upon Bike Rentals

2.2.1 Exploration of Categorical Variables

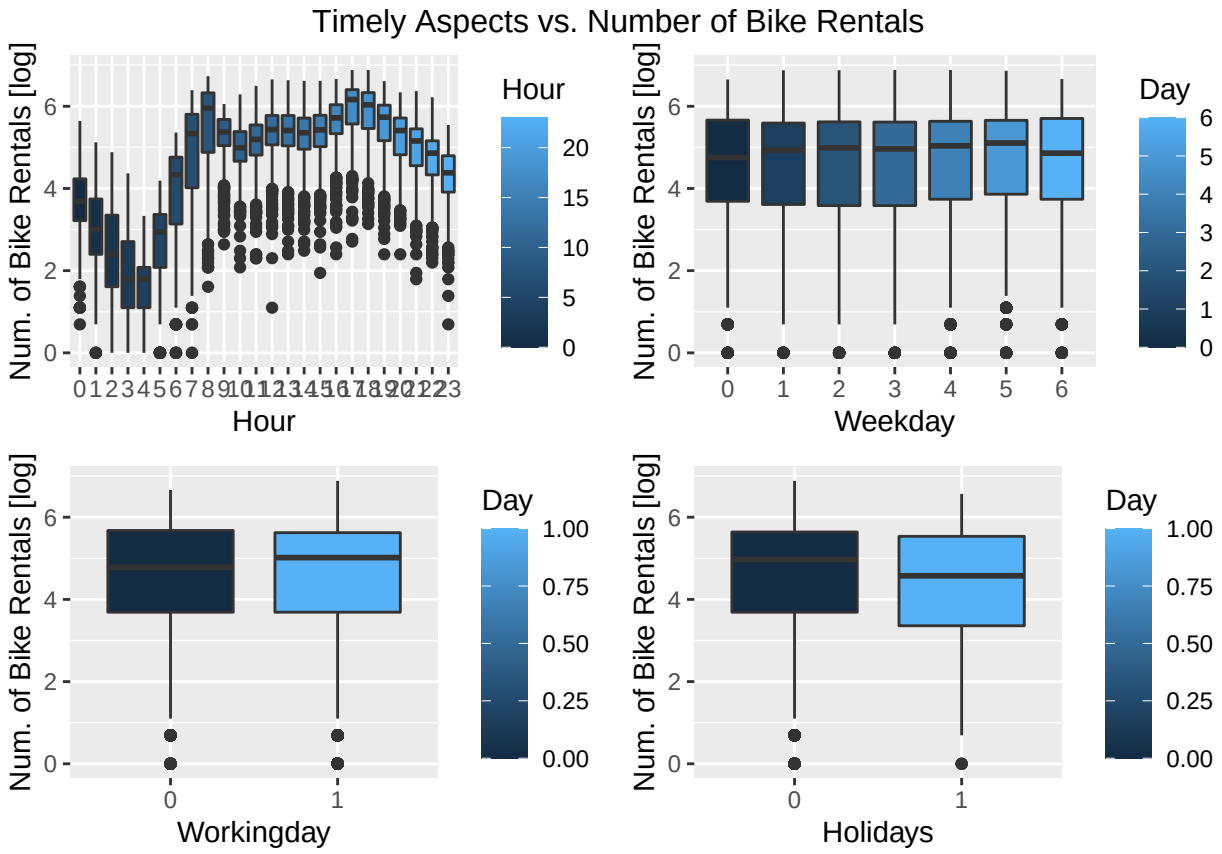
2.2.1.1 Seasonal Influence In order to investigate the seasonal influences, the monthly (1 to 12) and seasonal data are included in the dataset as shown below::

- Category 1: Spring
- Category 2: Summer
- Category 3: Fall
- Category 4: Winter



The two boxplot above indicate, that the mos bike rentals take place in summer and fall which we have expected.

2.2.1.2 Timely Influence In order to examine the temporal influences we have the time specification in hours (1 to 24), the weekdays (0 = Monday to 6 = Sunday), working day (1 = yes, 0 = no) as well as vacation day (1 = yes , 0 = no).



The above plots indicate, that the hour in which the bike is rented has the most influence of total rentals which seems obvious. On the other hand, whether there is a working day or not, seems not to have a significant influence.

In the next chapter we try to find a linear model which suits best the dateset.

3 Linear Regression Model

3.1 Data Preperation for the Linear Model

3.1.1 Set Categorical Variables as Factors

```
df$season <- factor(df$season, levels = c("1", "2", "3", "4"), ordered = FALSE)
df$yr <- as.factor(df$yr)
df$mnth <- as.factor(df$mnth)
df$hr <- as.factor(df$hr)
df$holiday <- as.factor(df$holiday)
df$weekday <- as.factor(df$weekday)
df$workingday <- as.factor(df$workingday)
df$weathersit <- as.factor(df$weathersit)
```

3.1.2 Re-Transform the Continious Weather Variables

As described in the data exploration chapter, the continuous weather variables have been normalized in the existing dataset. In order to make interpretation of the coefficients more straight forward in the linear model the variables will be re-transformed to the actual values as follows. For more advanced models the normalized variables may be the better choice to fit a model.

```
df$temperature <- (df$temp * (39 + 8)) - 8
df$atemperature <- (df$atemp * (39 + 8)) - 8
df$humidity <- (df$hum * 100)
df$wind <- (df$windspeed * 67)
```

3.1.3 Split data into train and test dataset

In the following the dataset will be split into a train and a test dataset. The partition will be used to evaluate the accuracy of the model when used on the test data.

```
set.seed(123)
indices <- createDataPartition(df$cnt, p=.8, list = F)
train <- df %>% slice(indices)
test <- df %>% slice(-indices)
```

3.2 Multiple Linear Regression Model with Continious Weather Predictors

First of all we want to include only the continuous variables in the model. As stated in the previous chapter we consider the response variable as amount data and therefore take the log of it to build the model. Indeed, if we take the log we get much better for all basic linear regression fits.

```
lm.fit.0.train <- lm(log(cnt) ~ temperature + atemperature + humidity + wind,data=train)
summary(lm.fit.0.train)
```

```
##
## Call:
## lm(formula = log(cnt) ~ temperature + atemperature + humidity +
##     wind, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5444 -0.6400  0.2519  0.8710  3.2934
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.9326457  0.0531309  92.839  < 2e-16 ***
## temperature   0.0029451  0.0080521   0.366    0.715
## atemperature  0.0640689  0.0090373   7.089 1.41e-12 ***
## humidity     -0.0234304  0.0005942 -39.429  < 2e-16 ***
## wind          0.0083912  0.0014307   5.865 4.59e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.29 on 13899 degrees of freedom
## Multiple R-squared:  0.2465, Adjusted R-squared:  0.2463
## F-statistic:  1137 on 4 and 13899 DF,  p-value: < 2.2e-16
```

3.2.1 Simplification of Model lm.fit.0.train

There is evidence that the predictor temperature has no significant influence since the p-value is large with 0.7. However the feels like temperature (atemperature) is a result of the combination of the temperature, the humidity and the wind at a given time. Moreover, the prediction of future bike rental will depend from the weather forecast. In the forecast the feels like temperature is not always available. Therefore we try to fit the model without atemperature to simplify it and make it more practical for future usage.

```r
lm.fit.1.train <- lm(log(cnt) ~ temperature + humidity + wind,data=train)
summary(lm.fit.1.train)
```

```
##
## Call:
## lm(formula = log(cnt) ~ temperature + humidity + wind, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5827 -0.6534  0.2505  0.8791  3.3242
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.0052296  0.0522274  95.835  < 2e-16 ***
## temperature  0.0593705  0.0012224  48.570  < 2e-16 ***
## humidity    -0.0232619  0.0005948 -39.108  < 2e-16 ***
## wind         0.0060989  0.0013961   4.368 1.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.292 on 13900 degrees of freedom
## Multiple R-squared:  0.2438, Adjusted R-squared:  0.2436
## F-statistic:  1494 on 3 and 13900 DF,  p-value: < 2.2e-16
```

Interpretation of Regression Coefficients: - As we can see from the output above the predictor temperature and wind have both a positive influence upon the number of bike rentals. A increase of one unit (temperature) leads to a increase of 5.9 % rentals and 0.6 % when the wind is increased by one unit. The humidity on the other hand has a negative influence. If the humidity is increased by one unit the number of rentals decreases by 2.3 %.

3.2.2 Accessing the model accuracy

We evaluate the accuracy of both models by testing them with the test data partition. The R-squared will serve as an indication of the accuracy as well as the RSME values.

3.2.2.1 Compare R-squared Values

```
## [1] "R-squared of lm.fit.0: 0.258"
```

```
## [1] "R-squared of lm.fit.1: 0.254"
```

- Both R-squared values are almost identical when applying the model on the test data. Therefore, the predictor atemperature shall be neglected for the linear model in the further model building.

3.2.2.2 Compare RMSE Values

```
## [1] "RMSE of lm.fit.0: 171.7"
```

```
## [1] "Percentage error of lm.fit.1: 91 %"
```

```
## [1] "RMSE of lm.fit.0: 173.5"
```

```
## [1] "Percentage error of lm.fit.1: 92 %"
```

- The RMSE values for both models are roughly 170 units as we can see from the output above. This corresponds to a percentage error of around 90 % of the predicted values if it is compared with the mean of all bike rentals.

3.3 Multiple Linear Regression Model with Continious and Categorical Variables

```
# Update the previous model lm.fit.1.train, including all the categorical variables.
lm.fit.2.train <- update(lm.fit.1.train,. ~ . + weathersit + hr + season + mnth + workingday + weekday +
# formula(lm.fit.2.train)
```

3.3.1 Simplification of Model lm.fit.2.train

In the next step we try fo find again a more simple and more practical model without reducing the predicting accuracy of the model significantly. With the drop1 function we test influence of the categorical variables with a F test as follows.

```
drop1(lm.fit.2.train, test = "F")
```

```
## Single term deletions
##
## Model:
## log(cnt) ~ temperature + humidity + wind + weathersit + hr +
##     season + mnth + workingday + weekday + holiday + yr
##             Df Sum of Sq     RSS      AIC  F value    Pr(>F)
## <none>                    5422.5 -12988.4
## temperature  1     199.6  5622.0 -12487.9  509.768 < 2.2e-16 ***
## humidity     1      16.4  5438.9 -12948.3   42.000 9.435e-11 ***
## wind         1      12.0  5434.4 -12959.7   30.564 3.288e-08 ***
## weathersit   3     275.1  5697.6 -12306.3  234.242 < 2.2e-16 ***
## hr          23   15615.1 21037.6   5816.1 1734.333 < 2.2e-16 ***
## season       3     137.5  5560.0 -12646.1  117.104 < 2.2e-16 ***
## mnth        11      46.6  5469.0 -12891.5   10.815 < 2.2e-16 ***
## workingday   0       0.0  5422.5 -12988.4
## weekday      5      48.4  5470.9 -12874.8   24.726 < 2.2e-16 ***
## holiday      0       0.0  5422.5 -12988.4
## yr           1     746.9  6169.4 -11196.1 1908.057 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is evidence that the variables workingday an holiday have no significant influence according the results of the F-test. We therefore neglect this variables in the model lm.fit.3.train. The graphical analysis in the last chapter supports this step, showing that there is no influence of these variables. We exclude these variables for the final model.

```
lm.fit.3.train <- update(lm.fit.1.train,. ~ . + weathersit + hr + season + mnth + weekday + yr ,data=tra
# formula(lm.fit.3.train)
```

3.3.2 Accessing the model accuracy

3.3.2.1 Compare R-Squared Values

Again we compare the R-squared values of the 3 models.

```
## [1] "R-squared of lm.fit.2: 0.828"
```

```
## [1] "R-squared of lm.fit.3: 0.828"
```

- Form the previous model the the most simplified we can see only a marginal drop of the R-squared values. Therefore, it can be stated that the simplified model can be used without losing considerable information.

3.3.2.2 Compare RMSE Values

```
## [1] "RMSE of lm.fit.2: 96.9"
```

```
## [1] "Percentage error of lm.fit.2: 51.4 %"

## [1] "RMSE of lm.fit.3: 97.4"

## [1] "Percentage error of lm.fit.3: 51.7 %"
```

- The RMSE values for both 3 models are between 97 t0 110 units as we can see from the output above. This corresponds to a percentage error of around 51 to 59 % of the predicted values if it is compared with the mean of all bike rentals.

```
##
## Call:
## lm(formula = formula(lm.fit.3.train), data = test)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.08270 -0.30012  0.03358  0.38687  2.45689
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.9158137  0.0870855  33.482  < 2e-16 ***
## temperature  0.0305188  0.0027727  11.007  < 2e-16 ***
## humidity    -0.0032520  0.0007683  -4.233 2.37e-05 ***
## wind        -0.0030975  0.0014167  -2.186 0.028852 *
## weathersit2  0.0076810  0.0264884   0.290 0.771853
## weathersit3 -0.6381223  0.0453287 -14.078  < 2e-16 ***
## hr1         -0.5711212  0.0760137  -7.513 7.31e-14 ***
## hr2         -1.1189292  0.0747507 -14.969  < 2e-16 ***
## hr3         -1.8115857  0.0770261 -23.519  < 2e-16 ***
## hr4         -2.1754243  0.0753524 -28.870  < 2e-16 ***
## hr5         -0.9880170  0.0740504 -13.343  < 2e-16 ***
## hr6          0.2699303  0.0733150   3.682 0.000235 ***
## hr7          1.1836421  0.0738777  16.022  < 2e-16 ***
## hr8          1.8113397  0.0745190  24.307  < 2e-16 ***
## hr9          1.5328528  0.0744233  20.596  < 2e-16 ***
## hr10         1.2074100  0.0772795  15.624  < 2e-16 ***
## hr11         1.3185236  0.0735631  17.924  < 2e-16 ***
## hr12         1.4677899  0.0777346  18.882  < 2e-16 ***
## hr13         1.5111936  0.0757884  19.940  < 2e-16 ***
## hr14         1.3687965  0.0787133  17.390  < 2e-16 ***
## hr15         1.3703310  0.0779888  17.571  < 2e-16 ***
## hr16         1.6749705  0.0772950  21.670  < 2e-16 ***
## hr17         2.0597478  0.0763519  26.977  < 2e-16 ***
## hr18         1.9987986  0.0763101  26.193  < 2e-16 ***
## hr19         1.7304255  0.0755409  22.907  < 2e-16 ***
## hr20         1.4462195  0.0745160  19.408  < 2e-16 ***
## hr21         1.1974803  0.0749683  15.973  < 2e-16 ***
## hr22         0.9688490  0.0767917  12.617  < 2e-16 ***
## hr23         0.5323629  0.0760673   6.999 3.10e-12 ***
## season2      0.3512380  0.0652387   5.384 7.78e-08 ***
## season3      0.3458522  0.0795131   4.350 1.40e-05 ***
## season4      0.6531951  0.0680516   9.599  < 2e-16 ***
## mnth2        0.1816174  0.0516889   3.514 0.000448 ***
## mnth3        0.1380711  0.0573184   2.409 0.016056 *
## mnth4        0.1125635  0.0885984   1.270 0.203996
## mnth5        0.2139644  0.0940304   2.275 0.022939 *
```

```
## mnth6        0.1131568  0.0969157   1.168 0.243058
## mnth7        0.0692150  0.1085678   0.638 0.523824
## mnth8        0.0770516  0.1063655   0.724 0.468867
## mnth9        0.1905547  0.0954845   1.996 0.046050 *
## mnth10       0.0906998  0.0896021   1.012 0.311490
## mnth11      -0.0944324  0.0856004  -1.103 0.270028
## mnth12      -0.0512931  0.0685889  -0.748 0.454612
## weekday1    -0.0307332  0.0391408  -0.785 0.432393
## weekday2    -0.0725483  0.0403478  -1.798 0.072254 .
## weekday3    -0.0026584  0.0399542  -0.067 0.946956
## weekday4     0.0108687  0.0399415   0.272 0.785549
## weekday5     0.1119988  0.0390190   2.870 0.004125 **
## weekday6     0.0745011  0.0387918   1.921 0.054873 .
## yr1          0.4902112  0.0215924  22.703  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6225 on 3425 degrees of freedom
## Multiple R-squared:  0.8278, Adjusted R-squared:  0.8254
## F-statistic: 336.1 on 49 and 3425 DF,  p-value: < 2.2e-16
```

- Note from the output above, that the p-values for some month are not significant. More on that on in the next chapter.

3.4 Summary

As summary we want to answer following 4 questions regarding the linear regression model and its interpretation.

3.4.1 Is there a relationship betweeen the independent variables and the response variable?

As stated above there is evidence, that the weather as well as the timely and seasonal variables has influence of the number of bike rentals according to the P-values for the continuous variables as well as the outcome of the F-test for the categorical variables.

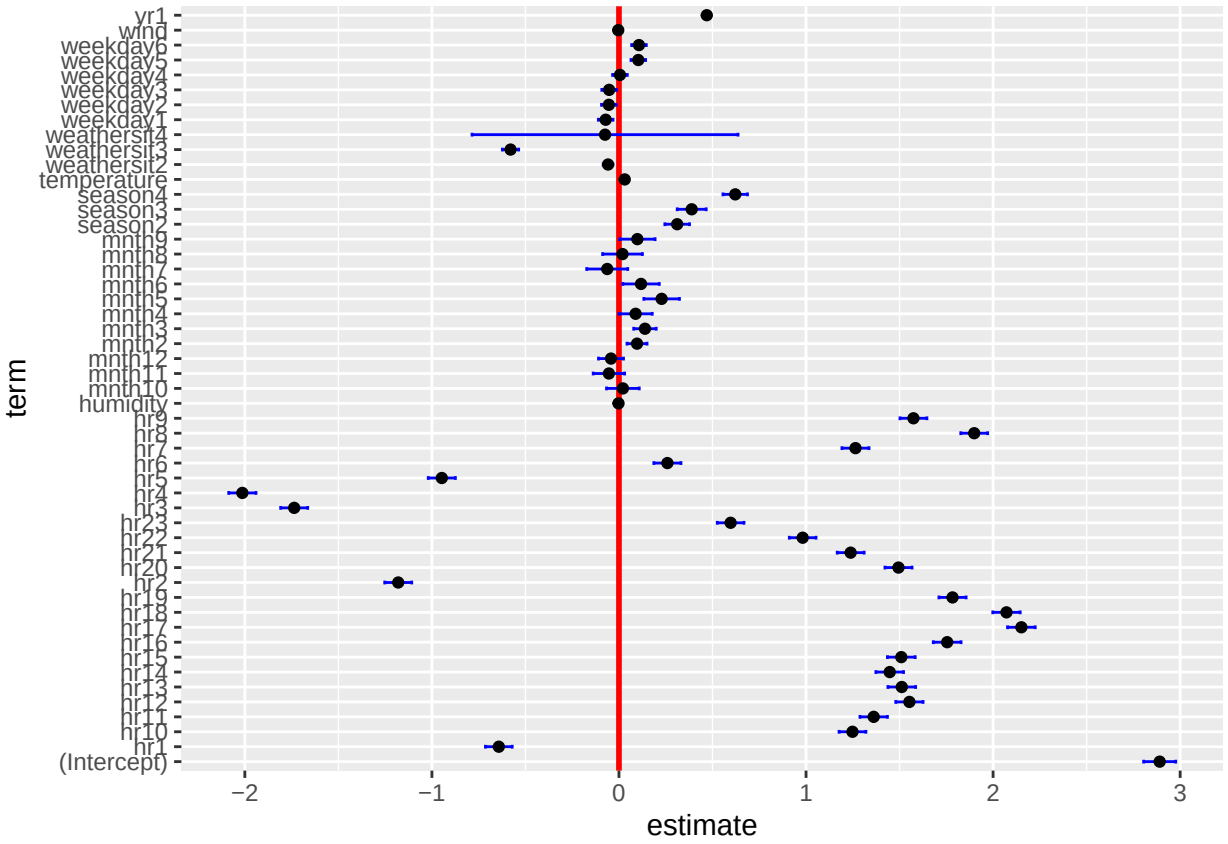
3.4.2 How strong is the relationship and how accurate is the model?

The predictors used in the model lm.fit.4 explain about 80 % of the variance in cnt (number of bike rentals) according to the R-squared values. If we look at the RMSE value of the lm.fit.3 we achieve a value of 97.4 which corresponds to a percentage error of around 51.7 %.

3.4.3 How large is the effect of each predctor on cnt?

To answer this question we will plot the confidence interval of all the predictors.

```
ggcoef(lm.fit.3.train,vline_color = "red", vline_linetype =  "solid", errorbar_color = "blue", errorbar
```



We can see that for the predictor month all dummy variables are crossing the zero line, indicating that this variable is not statistically significant. Therefore we could have dropped this predictor as well even though the p-values where low in the model examination. The evaluate this assumption we update the model and compare the two R-squared values.

```
## [1] 0.8278342
```

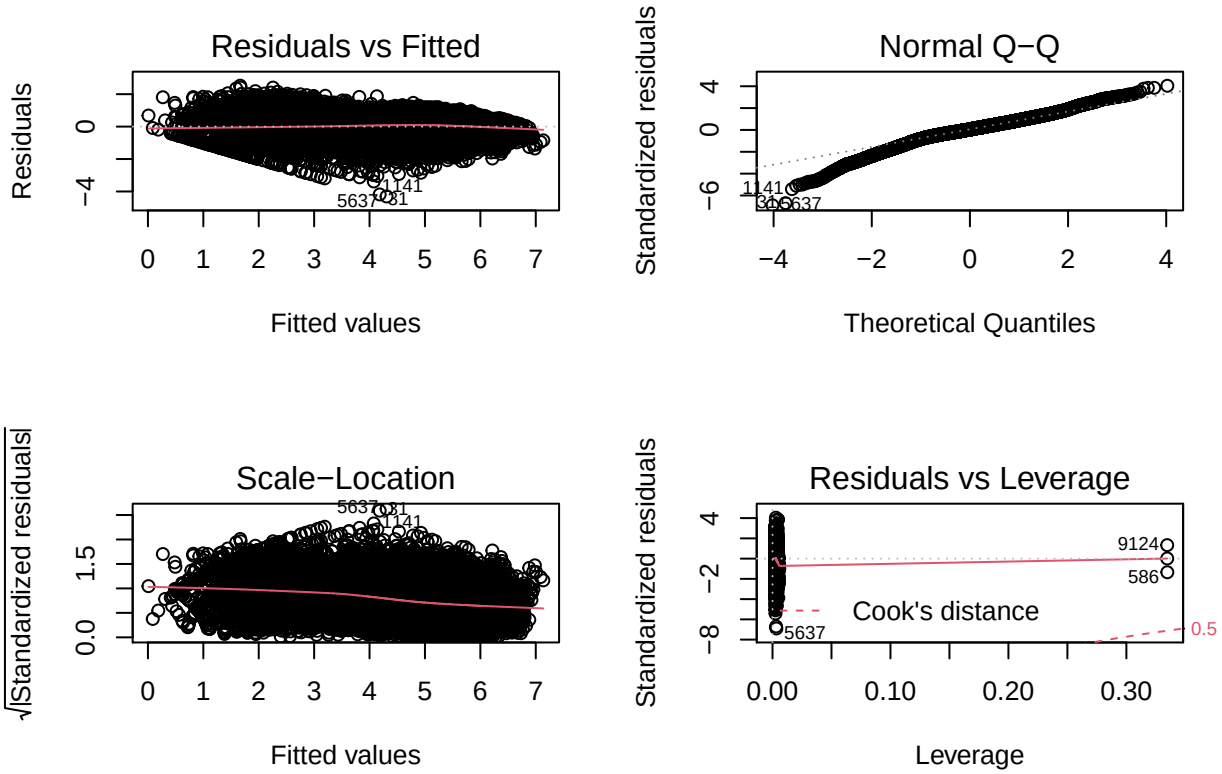
```
## [1] 0.7996572
```

In fact, the R-squared value decreases only marginally when the predictor month is not taken into account, even though the F-statistic has classified the predictor as significant. Finally, it can be concluded that the temperature as well as the current time have the greatest effect upon the number of bike rentals, since the confidence intervals are the furthest away from zero.

3.4.4 Are there Potential Problems of Fitting this data with a Linear Regression Model

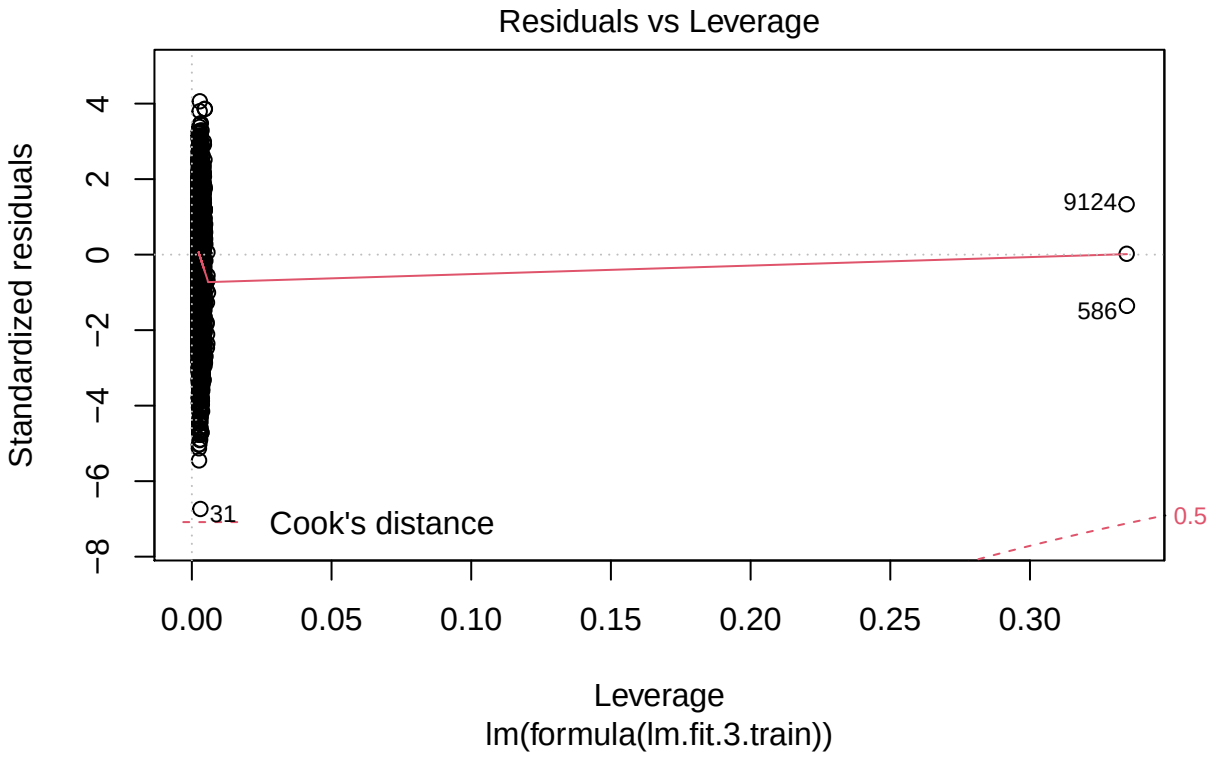
With the plot function, we try to examine our model in more detail by means of residual analysis.To to that we build a model with the entire dataset.

```
lm.fit.0.full = lm(formula(lm.fit.3.train),data=df)
par(mfrow=c(2,2))
plot(lm.fit.0.full)
```



- The residual vs fitted plot shows no clear pattern indicating that that relationship is linear. Also the smoother stays more or less on zero. In a clear non-linear situation this would have been the case.
- The QQ-plot shows that the residuals seem not to follow a normal distribution. This circumstance could be a sign that the linear model is not quite suitable to form a predictive model. We assume that this is due to the distribution of bike rentals. These are not normally distributed and could be considered as amount data.
- Checking the scale-location plot, it seems that the residuals are spread fairly randomly along the horizontal line. Although the variance seems to decrease on the right hand side of the plot we consider this as "normal".
- In the residual vs leverage, one observation seems to lie outside of the cooks distance. It is the observation 5637 in the dataset. A closer look reveals an error in the dataset. On 27.08.2011 and 28.08.2011 the time seems to be incorrect and therefore the observation 5637 can be considered as an outlier. It would probably make sense to delete these two days from the dataset. We do this in the following step and form a new model.

```
df2 <- df[!(df$dteday=="2011-08-27" | df$dteday=="2011-08-28"),]
lm.fit.1.full = lm(formula(lm.fit.3.train),data=df2)
plot(lm.fit.1.full, which = 5)
```




```
summary(lm.fit.0.full)$r.squared
```

```
## [1] 0.8234896
```

```
summary(lm.fit.1.full)$r.squared
```

```
## [1] 0.824626
```

- As we can see, all observations are lying now within the cook distance and the R-value also increases marginally when taking out the faulty date entries in a updated dataset.

In the next chapter we will use more advanced linear models to find a better fit for the data.

4 General Additive Model (GAM)

In this chapter we would like to use the GAM model to address for possible non-linear relationships and improve the basic linear mode lm.fit.3.

4.1 Check if model with non-linear function ist better

```
gam_0_train <- gam(log(cnt) ~ temperature + humidity + wind,data=train)
gam_1_train <- gam(log(cnt) ~ s(temperature) + humidity + wind,data=train)
gam_2_train <- gam(log(cnt) ~ temperature + s(humidity) + wind,data=train)
gam_3_train <- gam(log(cnt) ~ temperature + humidity + s(wind),data=train)

anova(gam_0_train,gam_1_train,gam_2_train,gam_3_train,test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: log(cnt) ~ temperature + humidity + wind
## Model 2: log(cnt) ~ s(temperature) + humidity + wind
```

```
## Model 3: log(cnt) ~ temperature + s(humidity) + wind
## Model 4: log(cnt) ~ temperature + humidity + s(wind)
##   Resid. Df Resid. Dev       Df Deviance        F    Pr(>F)
## 1     13900      23196
## 2     13892      22737  7.96329   458.54   35.183 < 2.2e-16 ***
## 3     13892      22978 -0.10501  -240.84 1401.296 < 2.2e-16 ***
## 4     13895      23161 -3.15690  -182.71   35.363 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- There is strong evidence that for all models applying the smoothing splines for the 3 continuous weather predictors is needed to improve the model. Consequently we will use the model applying the smoothing splines on all 3 weather predictor.

4.1.1 Compare RMSE Values

```
## [1] "RMSE of gam_01(simple linear model): 173.5"
```

```
## [1] "RMSE of gam_02(s(temperature)): 166.9"
```

```
## [1] "RMSE of gam_02(s(humidity)): 171.6"
```

```
## [1] "RMSE of gam_02(s(wind)): 172.9"
```

```
## [1] "RMSE of gam_02(s(on all variables)): 164.1"
```

- from the output above we can conclude that for all variables we get an improvement of the RMSE value when applying smoothing splines on it. Therefore we conclude that the relationship between predictors and response variable is not linear.
- for the gam_01 model we get the same result as for the lm.fit.1 model since it is nothing else than a simple linear model without any non linear functions in the gam() function.

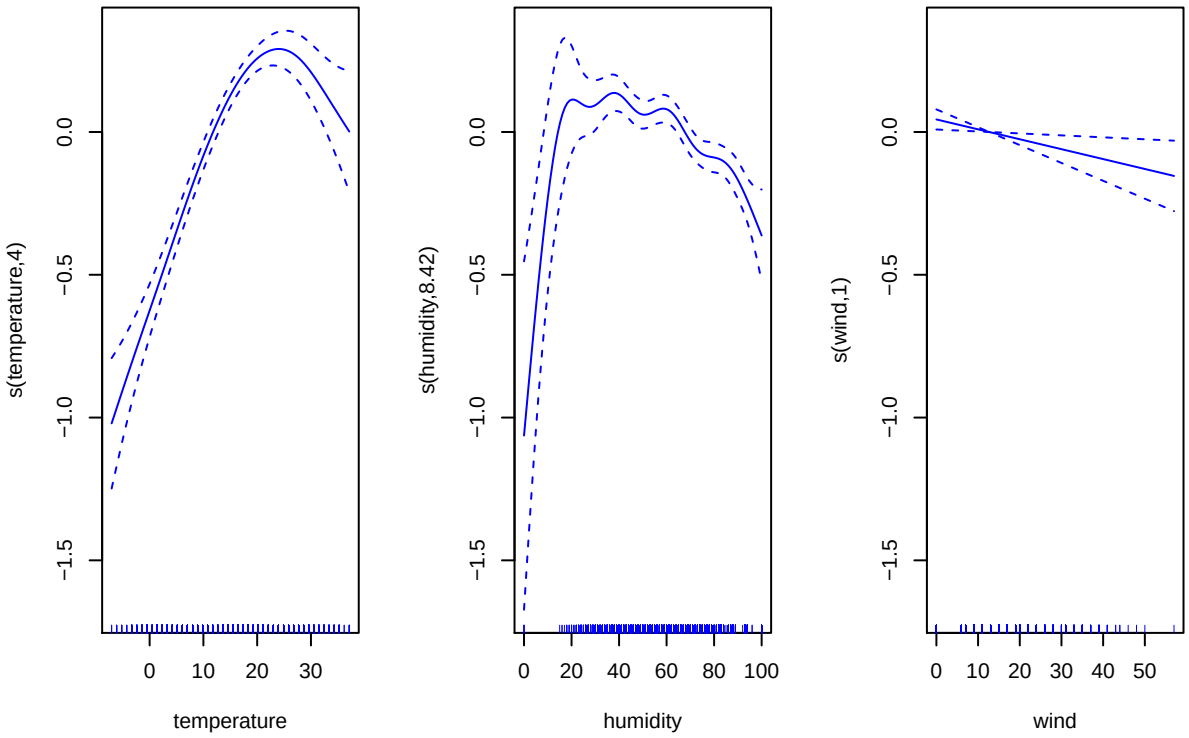
4.2 Add categorical variables to GAM Model gam_04

```
## [1] "RMSE of gam_5: 90.4"
```

- We get a RSME of 90.4. Comparing to the lm.fit.3 with a RSME 97.4 we have an improvement of roughly 7.5 %.
- Below for the plots with the smoothing splines for each variable with and without the residuals.

4.3 Summary

- It can be stated that using a GAA model with smoothing splines improves the model compared to a basic linear model. We have tried to add smoothing splines to the categorical variables but somehow we were not able to run this function. This could have improved the model even more.



5 GLM Binomial and Poisson

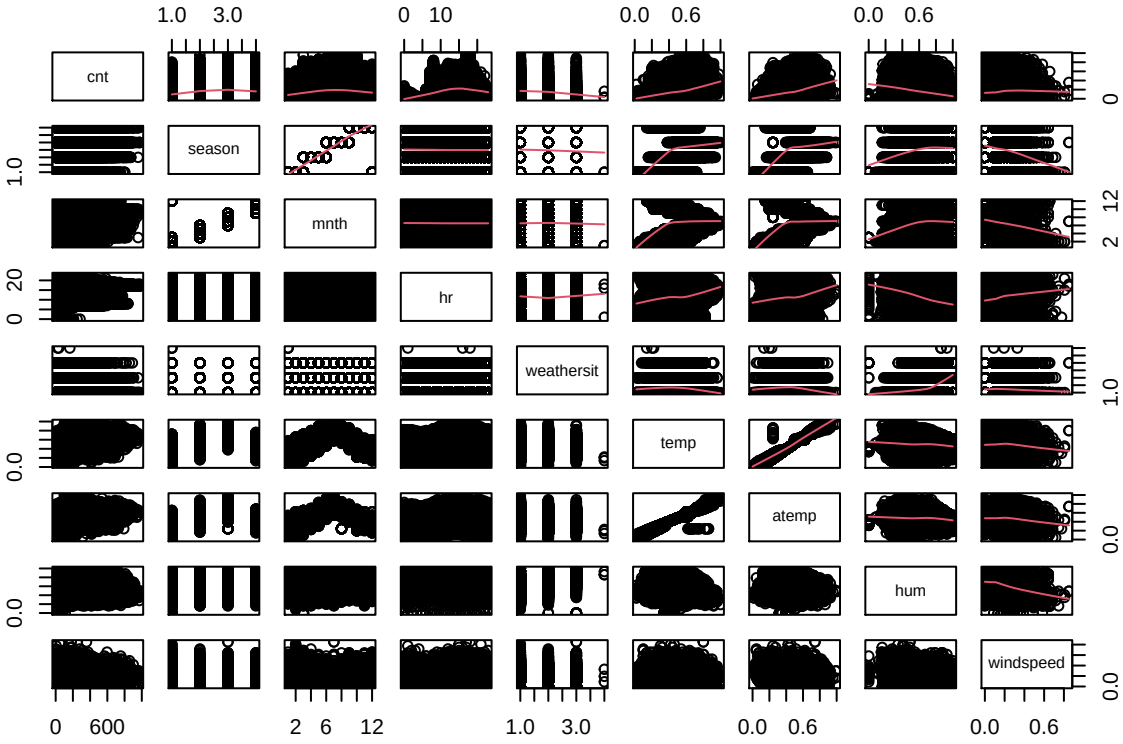
Reading in the daily data as "day" and the hourly data as "hour"

For this model we are also going to drop the data from the 27th and 28th of August 2011, for the reasons shown in the last chapter.

5.1 Plotting and inspecting the data

The variable for casual or registered users is less of importance here, as it does not necessarily help with the prediction of the bike demand. Especially as we don't see drastically different behaviour regarding usage. Just from the graph above one could assume, that it is slightly more likely that registered users use the bikes also in more extreme weather situations (hot, cold, windy, etc.). Furthermore, we will not use the type of users (registered or casual) as predictors, as they make up our main response variable (count) and therefore would lead to highly fitting models but with no value for prediction.

So let's have another look at the following variables: - season - mnth - hour - weathersit - temp - atemp - hum - windspeed



5.2 Creating training and testing sets

First we divide the data into a training and a testing set. We use 80% of the data to train the model and 20% to test it. Furthermore we divide the testing set into one we use for the prediction and one to check those predictions.

```
set.seed(123)
indices.day <- createDataPartition(day$cnt, p = .8, list = F)
day.train <- day %>% slice(indices.day)
day.test_in <- day %>% slice(-indices.day) %>% select(-cnt)
# contains everything except the count values
day.test_truth <- day %>% slice(-indices.day) %>% pull(cnt)
# contains the TRUE count values of the testing set


set.seed(123)
indices.hour <- createDataPartition(hour$cnt, p = .8, list = F)
hour.train <- hour %>% slice(indices.hour)
hour.test_in <- hour %>% slice(-indices.hour) %>% select(-cnt)
hour.test_truth <- hour %>% slice(-indices.hour) %>% pull(cnt)
```

5.3 Creating a binomial model

First we have a look at a simple quasibinomial model, with the assumption that there are no interactions and all effects are linear. As we only have two binary variables, we are only going to check those two here. We use ilogit() as we need the count numbers as a value between 0 and 1.

```
##
## Call:
## glm(formula = ilogit(cnt) ~ season + yr + mnth + hr + holiday +
##     weekday + workingday + weathersit + temp + atemp + hum +
```

```
##     windspeed, family = "quasibinomial", data = hour)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.46292   0.01258   0.02976   0.07352   0.66357
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.611690   0.169692  15.391  < 2e-16 ***
## season       0.285532   0.039099   7.303 2.94e-13 ***
## yr           0.479900   0.055221   8.691  < 2e-16 ***
## mnth         0.018831   0.010570   1.782  0.07485 .
## hr           0.244211   0.007213  33.855  < 2e-16 ***
## holiday     -0.113031   0.186268  -0.607  0.54398
## weekday      0.065357   0.014567   4.487 7.28e-06 ***
## workingday  -0.678085   0.066569 -10.186  < 2e-16 ***
## weathersit  -0.123332   0.043167  -2.857  0.00428 **
## temp         5.529369   1.400099   3.949 7.87e-05 ***
## atemp       -0.900887   1.532544  -0.588  0.55665
## hum         -1.340083   0.189082  -7.087 1.42e-12 ***
## windspeed   -1.266559   0.310608  -4.078 4.57e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.0555428)
##
##     Null deviance: 568.71  on 17044  degrees of freedom
## Residual deviance: 337.87  on 17032  degrees of freedom
##   (299 observations deleted due to missingness)
## AIC: NA
##
## Number of Fisher Scoring iterations: 10
```

We decided to take out the holidays, as there are only very few anyways, the assumed temperature (atemp) as it is very close to the temperature (temp) and also dependent on the temperature, humidity and wind speed. We also took out the year. Instead we added interaction for the weather situation and the three weather variables temperature, wind speed and humidity.

It looks like the weather situation does not have a significant impact and neither does wind speed or the interaction of the weather situation and the wind speed. All the other predictors show a significance. So we have a another look at it without the weather situation and wind speed.

5.3.1 Fitting the binomial model with "train" data

For the binomial model we use the quasibinomial family, as the data is overdispersed and we do not use a binary, but a binomial response variable.

And have a look at the quantiles:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9740  0.9895  0.9949  0.9950  1.0013  1.0144
```

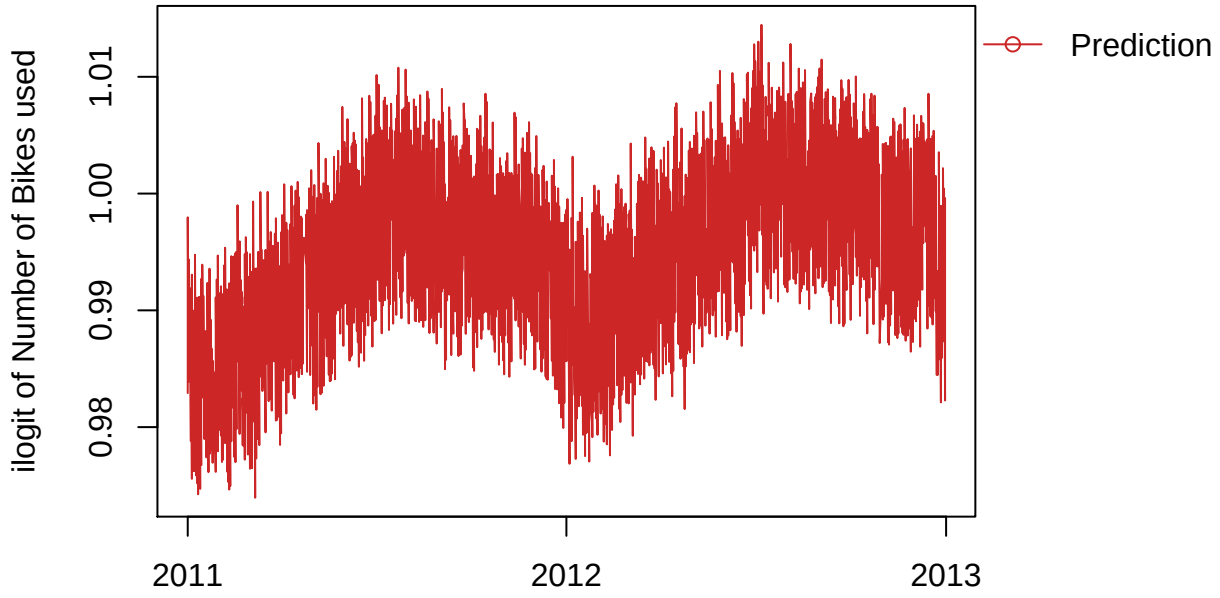
Checking root mean square error (RMSE)

```
## [1] NaN
```

It looks like there are missing values (probably through applying the ilogit() function) and therefore the RMSE could not be calculated. Furthermore the ilogit() function will lead to a different result with different

values and therefore cannot produce a useful result here.

So let us just have look at a visualisation of our predicted values.



The plot of the model prediction looks like it could work, but checking it with the true data is proving difficult. So we try a new approach.

To fit a binomial model we will try to predict the proportion of registered users versus casual users. For this we use the percentage of registered users out of the total number of users as the response variable.

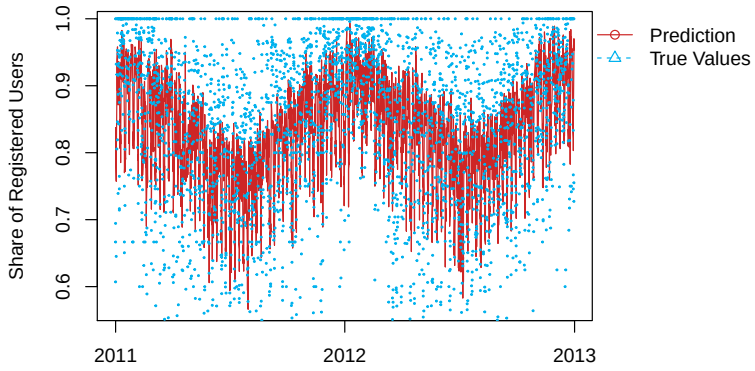
```
glm.binomial.2 <- glm((registered/cnt) ~ season + yr + mnth + hr + holiday + weekday + workingday + wea

glm.binomial.train.2 <- glm(formula = formula(glm.binomial.2), data = hour.train)
```

The following predictors seems to show significance: - Season - Year - Hour - Weekday - Workingday - Temperature - Humidity

We did not check for interactions as we already saw that there are interactions between the weather situation and the weather variables.

Making a prediction based on the test data and looking at the quantiles.

Visualisation of the second binomial model:



As the hourly data varies a lot, it is difficult to see if the model matches or not.

Checking root mean square error (RMSE)

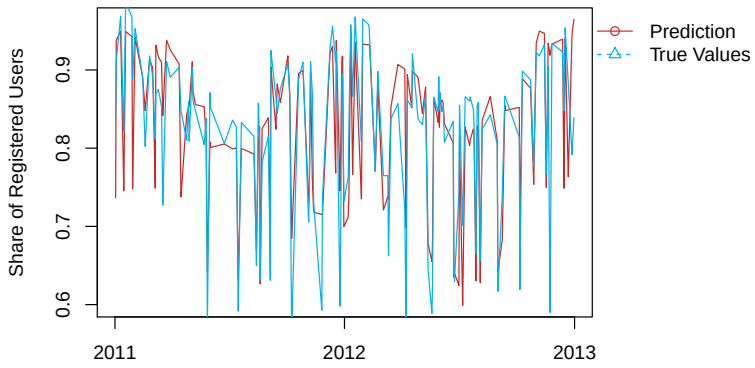
```
## [1] 0.1105669
```

With a RMSE of about 11 percentage points, the model does not look too bad.

But let's try to fit the model with daily data to bring down the variance.

Making a prediction based on the test data:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5989  0.7486  0.8403  0.8228  0.9030  0.9647
```



Here we see our prediction in red and the true values in blue.

Checking root mean square error (RMSE)

```
## [1] 0.06111774
```

We see that our model fitted on daily data became more accurate. With a RMSE of roughly 6 percentage points this seems like a pretty good fit, regarding the ratio depends on human decision (being registered or casual and using a bike or not).

Now we try the model with only the significant predictors from above (summary(glm.binomial.3)). Additionally we drop the wind speed, as this is usually less available data than temperature.

```
glm.binomial.4 <- glm((registered/cnt) ~ season + workingday + weathersit + temp, data=day, family="quas

glm.binomial.train.4 <- glm(formula = formula(glm.binomial.4), data = day.train)
```

Making a prediction based on the test data:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##  0.6034  0.7487  0.8416  0.8228  0.9052  0.9582
```

Checking root mean square error (RMSE)

```
## [1] 0.06183254
```

This seems to have made the model slightly less accurate (RMSE of 0.06183254 compared to 0.0612635 from before). But the difference seems to be minimal and relies on data which is broader available. Together with the result from the summary above (summary(glm.binomial.4)) the hypothesis could be made, that registered users make up a bigger part of the total count of users, when the temperature is colder and on working days compared to warmer temperatures and non-working days. Or that casual users, in comparison to registered users, predominantly use the bike service on warmer days and during their spare time.

Let's have a look at the temperature and working day variable, which we assume has shown the greatest significance (see summary(glm.binomial.4))

```
exp(coef(glm.binomial.4)["temp"])
```

```
##      temp
## 0.1613712
```

A change in one temperature unit (as it is normalised in this data frame) leads to a change of roughly the factor of 0.165.

```
exp(coef(glm.binomial.4)["workingday"])
```

```
## workingday
##   3.081471
```

On working days we see the percentage of users being registered 3.08 times higher then on non-working days.

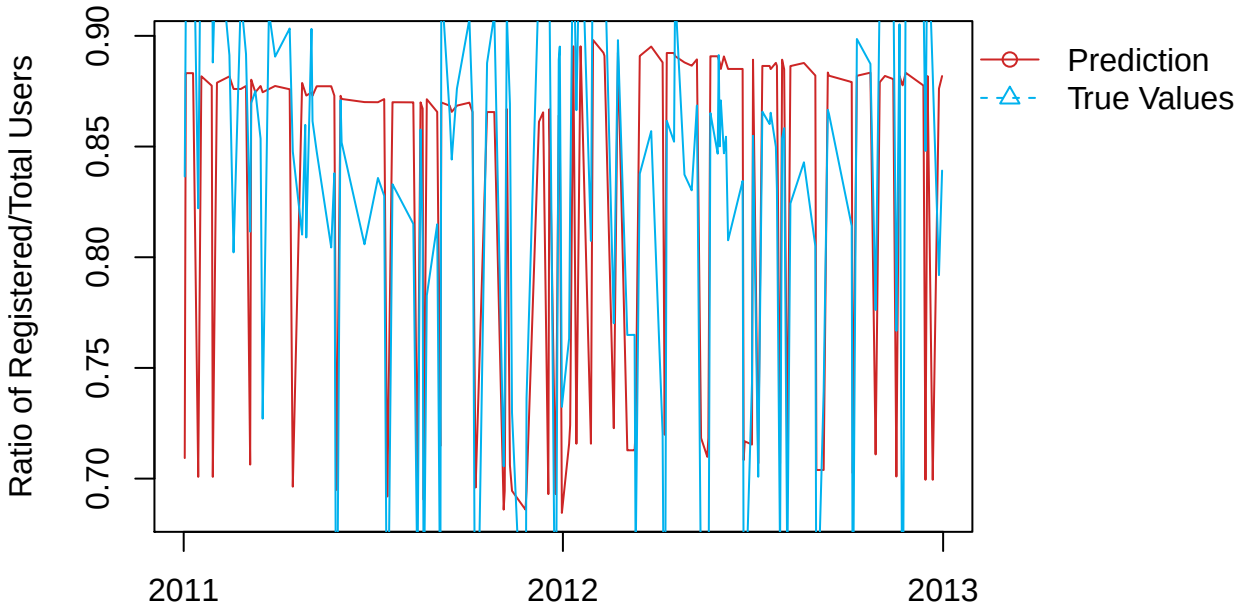
To make sure we are not missing anything, we will also fit a model to the date variables to compare. We still will keep in the working day variable, as this is not a weather variable but still was included in the former model.

```
glm.binomial.5 <- glm((registered/cnt) ~ yr + mnth + holiday + weekday + workingday, data=day, family="

glm.binomial.train.5 <- glm(formula = formula(glm.binomial.5), data = day.train)
```

Here we already see, that the working day is assumed as the only highly significant predictor in this model with year and month only showing some significance.

Making a prediction based on the test data and looking at the quantiles:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6845  0.7157  0.8730  0.8250  0.8833  0.8981
```



In this visualisation we can already see that the the model has a worse fit.

Checking root mean square error (RMSE)

```
## [1] 0.07468435
```

Surprisingly, the RMSE is not that much higher than from our predicted.binomial.test.4.

5.3.2 Conclusion: GLM Binomial

We would clearly prefer the third or fourth model here. With the fourth model being a little less accurate, but also needing less predictors, which might make the handling easier.

5.4 Creating a Poisson Model

We are using the family quasipoisson because of the overdispersion.

```
##
## Call:
## glm(formula = cnt ~ season + yr + mnth + hr + holiday + weekday +
##     workingday + weathersit + temp + atemp + hum + windspeed,
##     family = "quasipoisson", data = hour)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -30.293   -8.744   -3.026    3.958   38.699
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.7278480  0.0411563  90.578  < 2e-16 ***
## season       0.1166107  0.0114098  10.220  < 2e-16 ***
## yr           0.4387226  0.0118968  36.877  < 2e-16 ***
## mnth         0.0070461  0.0037213   1.893 0.058315 .
## hr           0.0457036  0.0009434  48.444  < 2e-16 ***
## holiday     -0.1324697  0.0388734  -3.408 0.000657 ***
```

```
## weekday      0.0080989  0.0029132   2.780 0.005441 **
## workingday   0.0201006  0.0128712   1.562 0.118384
## weathersit  -0.0198213  0.0109261  -1.814 0.069676 .
## temp         0.0393104  0.1874734   0.210 0.833916
## atemp        1.6566837  0.2123567   7.801 6.47e-15 ***
## hum         -1.0129528  0.0370093 -27.370  < 2e-16 ***
## windspeed    0.3135564  0.0499115   6.282 3.42e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 108.0685)
##
##     Null deviance: 2886027  on 17343  degrees of freedom
## Residual deviance: 1683021  on 17331  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Here it looks like quite some factors seem to be significant for the number of bikes used. With the temperature and the assumed temperature being very similar, we drop the assumed temperature from the model, with the measured temperature being more tangible. we also drop the weather situation as we still keep all the other weather variables. Furthermore, we drop the dates.

```
glm.pois2.hour <- glm(cnt ~ season + yr + mnth + hr+ holiday + weekday + workingday + temp + hum + winds

summary(glm.pois2.hour)
```

5.4.1 Fitting the poisson model with "train" data

We start with using all the variables except for the dates.

Making a prediction based on the test data:

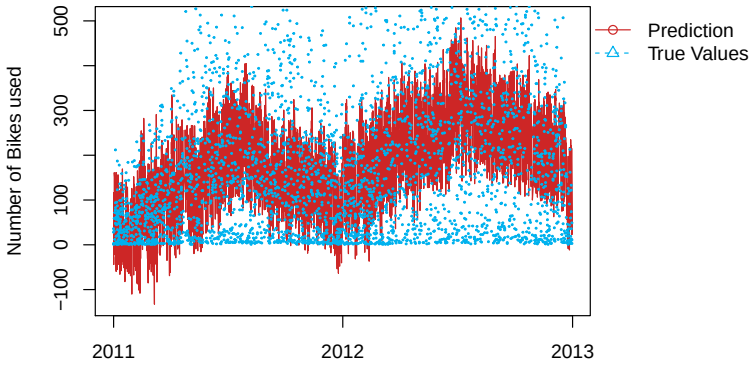
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -132.7   103.9   185.5   187.1   272.7   506.4
```

Note: Although we see a minimum of -132.7 here, in reality the number cannot go beneath 0.

Checking the root mean square error

```
## [1] 139.6343
```

Having a root mean square error (RMSE) of 139 when we have a median of 185.5 does definitely not sound very good. But we have to keep in mind, that it is based on hourly data. Let's have a look at a visualization of our prediction and the true data.



Here we see our predictions as a red line

and the actual numbers in blue dots. As hourly numbers vary strongly, we get quite some variance. Now we want to see if this model would fit better on a daily basis.

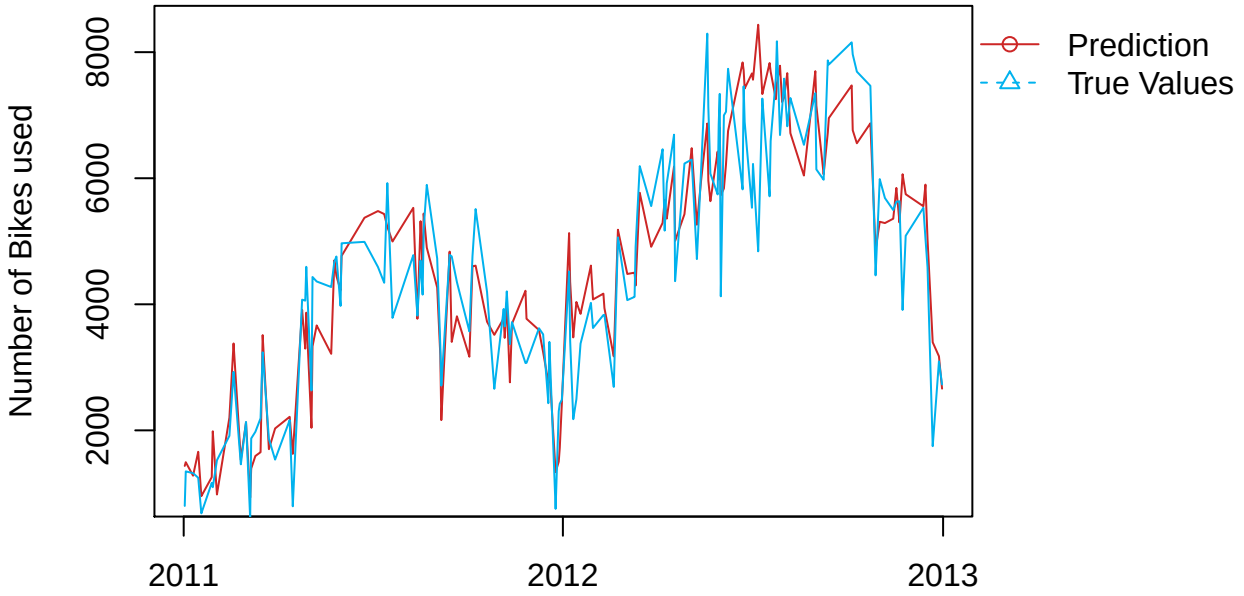
Making prediction on the test data

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   932.4  3323.6  4655.9  4600.7  5958.1  8434.9
```

Checking RMSE

```
## [1] 809.019
```

Here we have a RMSE of 809, which is higher than above in total. But as we are looking at daily data here, with a median of 4655.9 and numbers up to above 8000, this seems to be a better fit.



When looking at the predictions (red) in comparison to the true data (blue), this model does look usable if one wants to plan when to take bikes out of circulation for maintenance or repairs.

To see, if a variable is not needed, the model was tested by dropping only one of the predictors for each run through (with daily and hourly data). But the best result was achieved by using all the predictors (except for the type of users, for the reasons mentioned at the beginning of this chapter as well as the specific dates).

For practical purposes we have a look at a model with less weather data:

```
glm.poisson.3 <- glm(cnt ~ season + yr + mnth + holiday + weekday + workingday + weathersit + temp, data
glm.poisson.train.3 <- glm(formula = formula(glm.poisson.3), data = day.train)
summary(glm.poisson.3)
```

Making prediction on the test data

```
predicted.poisson.test.3 <- predict(glm.poisson.train.3, newdata = day.test_in)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   935.2  3388.7  4703.3  4621.5  5913.1  8327.3
```

Checking RMSE

```
## [1] 826.4995
```

Here we have a RMSE of 826, which is higher than it was for the second model. But as we need less variables and the weather situation and temperature seem to be the overall most important weather variables here, while being the most readily available ones, this model might be more practical.

5.4.2 Conclusion: GLM Poisson

While the second poisson model is more accurate, for practical reasons the third one might be more useful in this case, as it is still accurate enough so it would help with bike maintenance planning.

6 Support Vector Machine

6.1 Load the data

```
dt <- fread("../Data/hour.csv")
```

6.2 Prepare the data

The columns detday and instant are of no use. detday is represented in the variables yr, mnth, hr. Instant is only the record index. Therefore they are removed. the target variable "cnt" is moved to the first position. As already mentioned in section before, the data at the 27 and 28 August is not complete and wrong. Therefore, we will drop it for the SVM as well.

```
#drop the data for the 27 and 28 August 2011

dt <- subset(dt, dt$dteday != '2011-08-27')
dt <- subset(dt, dt$dteday != '2011-08-28')



#remove instant, dteday, casual and registered
dt <- dt %>% select(-c(1,2, 15, 16))



#move cnt to the first position
dt <- dt %>% select(cnt, everything())
```

6.3 Factor ordinal variables

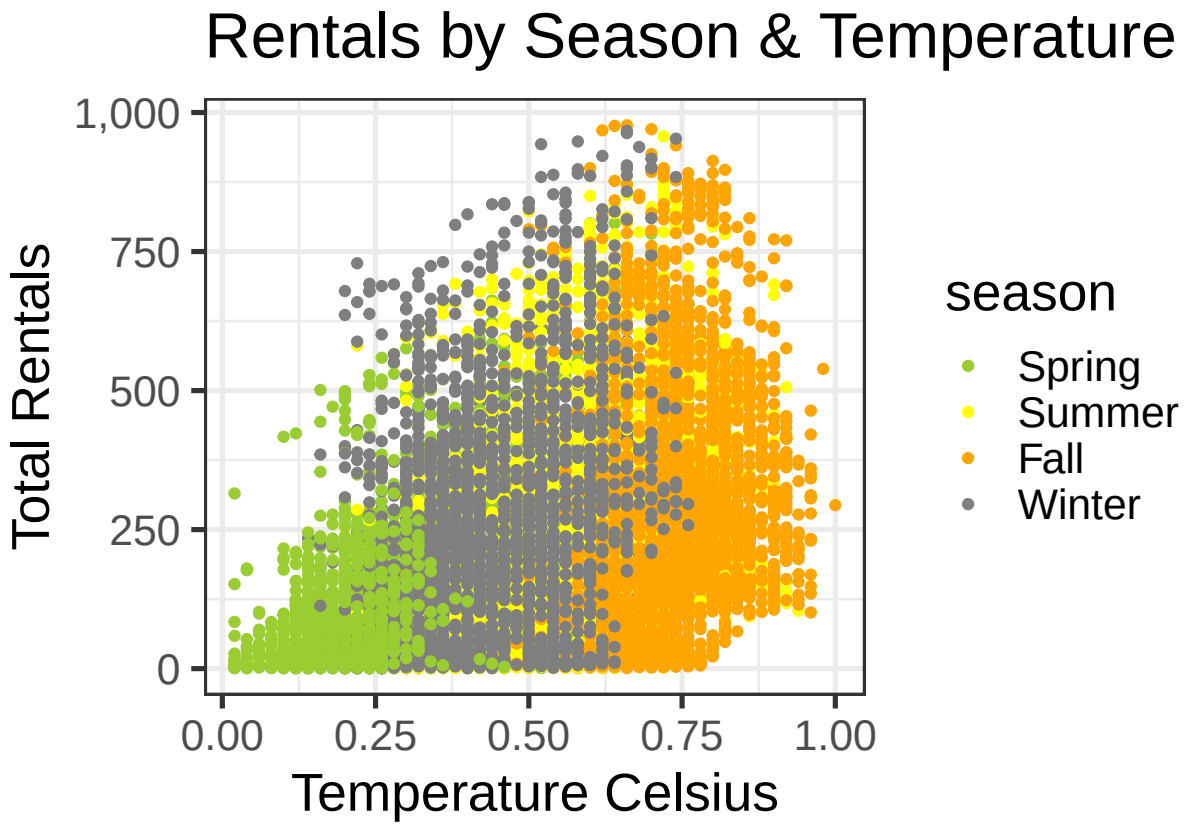
```
# Season
dt$season <- factor(
  dt$season, levels = c(1,2,3,4),
  labels = c('Spring', 'Summer', 'Fall','Winter'),
  ordered = TRUE)

# Year
dt$yr <- factor(dt$yr,
                levels = c(0, 1),
                labels = c(2011, 2012),
                ordered = TRUE)

# Month
dt$mnth <- factor(
```

```
  dt$mnth, levels = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
  labels = c('January','February', 'March', 'April',
             'May', 'June', 'July', 'August',
             'September','October', 'November', 'December'),
  ordered = TRUE)

# Hour
dt$hr <- as.factor(dt$hr)

# Holiday
dt$holiday <- factor(dt$holiday,
                     levels = c(0, 1),
                     labels = c('Workday', 'Weekend'))

# Weekday
dt$weekday <- factor(
 dt$weekday,levels = c(1,2,3,4,5,6,0),
  labels = c('Monday','Tuesday','Wednesday','Thursday',
             'Friday', 'Saturday','Sunday'),
  ordered = TRUE)

# Workingday
dt$workingday <- factor(dt$workingday,
                        levels = c(0,1),
                        labels = c('Workday', 'Weekend'))

# Weathersit
dt$weathersit <- factor(
  dt$weathersit,
  levels = c(1,2,3,4),
  labels = c('Clear, Few clouds, Partly cloudy, Partly cloudy', 'Mist+Clouds',
             'Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist', 'Light Snow, Light Rain + T

#dt %>% glimpse()
```

6.4 Plot data

6.4.1 Rentals by Season and Temperature

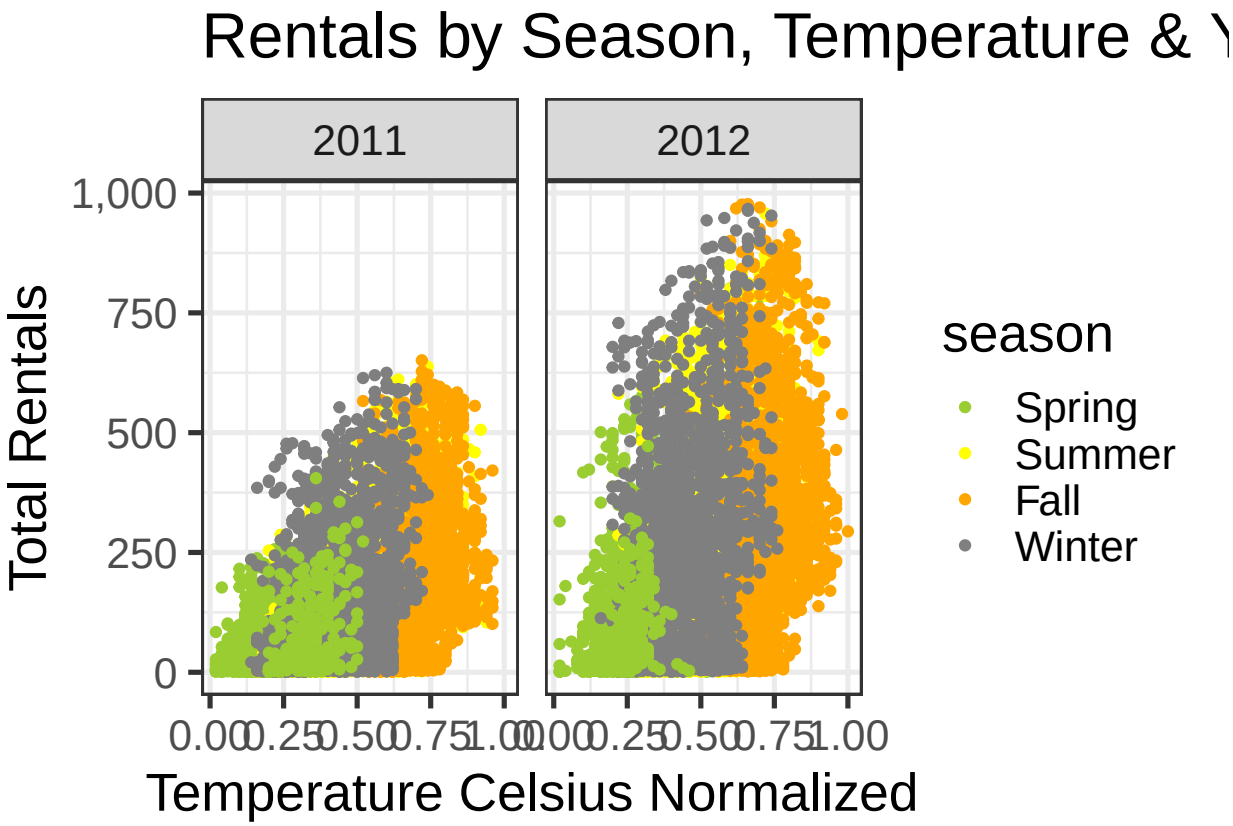
```
colr = c("olivedrab3", 'yellow', 'orange', 'grey50')
# Rentals by Season & Temperature
options(repr.plot.width=12, repr.plot.height=8)
ggplot(dt, aes(temp, cnt, color = season)) + geom_point() +
  theme_bw(base_size = 20) + scale_color_manual(values = colr) +
  labs(title = "Rentals by Season & Temperature", x = "Temperature Celsius", y = "Total Rentals") +
  scale_y_continuous(labels = scales::label_comma())
```



Interestingly, in this plot, the number of total rentals for spring is lower then for winter. But also the temperature when the rentals happen has more variablity than during the spring. The Temperature Celsius is normalized, therefore it ranges between 0 and 1.

6.4.2 Rentals by Season, Temperature and Year

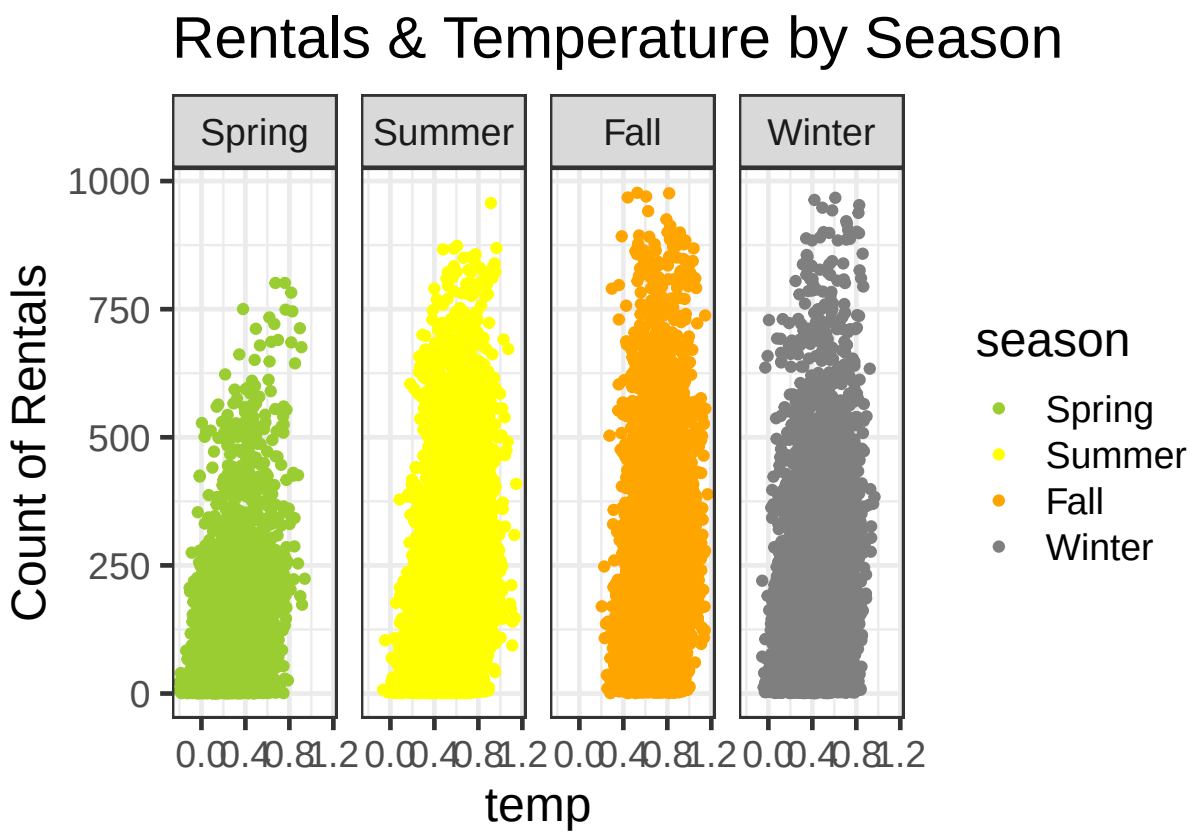
```
#Rentals by Season & Temperature & Year
ggplot(dt, aes(temp, cnt, color = season)) + geom_point() +
  theme_bw(base_size = 20) + scale_color_manual(values = colr) +
  labs(title = "Rentals by Season, Temperature & Year", x = "Temperature Celsius Normalized", y = "Total
  scale_y_continuous(labels = scales::label_comma()) +
  facet_grid(~yr)
```



It looks like there is an increasing trend between the total rentals from 2011 to 2012. We do not know if this trend continues.

6.4.3 Rental and Temperature by Seasion

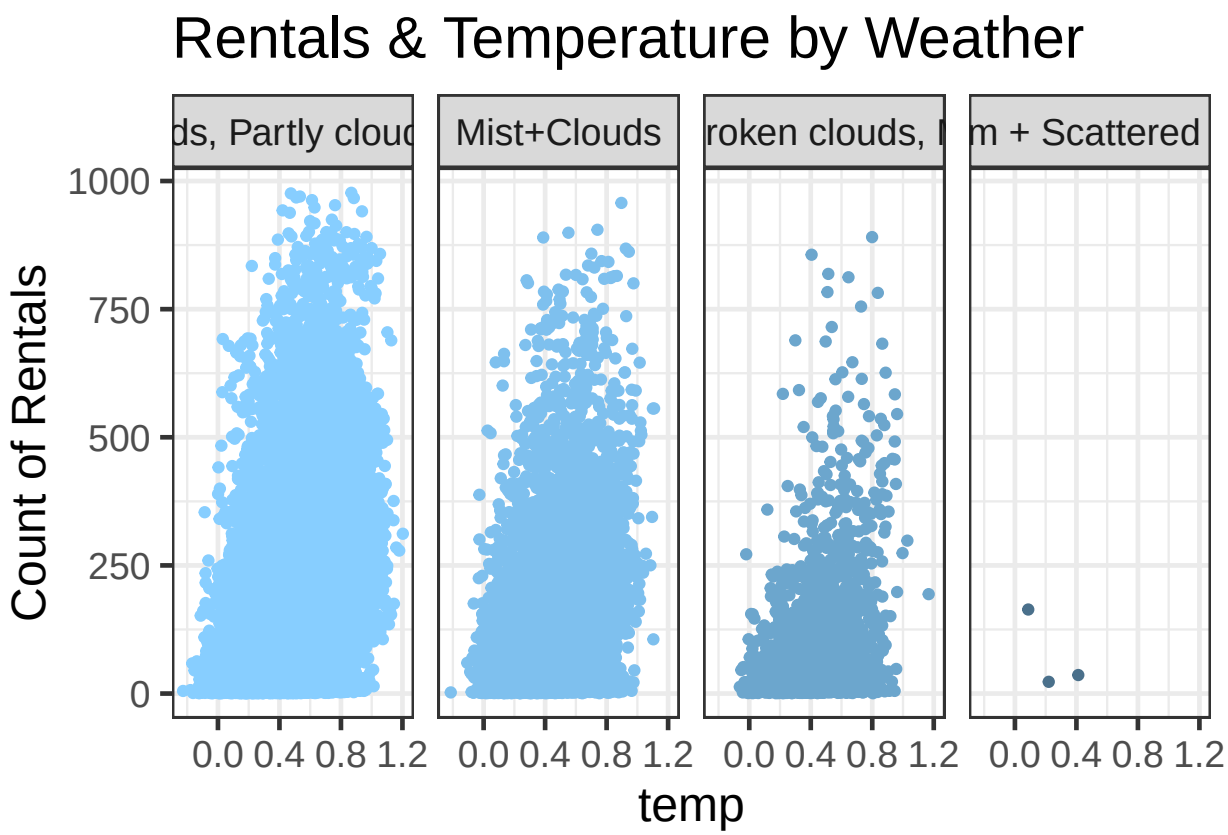
```
ggplot(dt, aes(temp, cnt, color = season)) +
  geom_jitter(width = 0.25) + scale_color_manual(values = colr) +
  labs(y="Count of Rentals", title = "Rentals & Temperature by Season") +
  facet_grid(~season) + theme_bw(base_size = 18)
```



Spring has clearly the lowest number of total rentals.

6.4.4 Rentals and Temperature by Weather

```
w = c('skyblue1','skyblue2','skyblue3','skyblue4','grey40')
ggplot(dt, aes(temp, cnt, color = weathersit)) +
  geom_jitter(width = 0.25, show.legend = F) +
  scale_color_manual(values = w) +
  labs(y="Count of Rentals", title = "Rentals & Temperature by Weather") +
  facet_grid(~weathersit) + theme_bw(base_size = 18)
```



When looking at different weather condition, from the best weather on the left to the worst weather on the right. The most rentals clearly happen when the weather is good. During the worst weather, almost no rentals happen. The weather seems to play a bigger role then the temperature.

##Multicollinearity Check for multicollinearity between the different response variables

```
m <- dt
# make all data types numeric
cols <- c("season", "weathersit", "workingday", "holiday", "mnth",
       "hr", "weekday","yr")
m[,cols] <- m %>% select(all_of(cols)) %>% lapply(as.numeric)

#checking multicolinearity with VIF
fit1 <- lm(cnt ~., data = m)
summary(fit1)
```

```
##
## Call:
## lm(formula = cnt ~ ., data = m)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -385.33  -93.27  -27.70   60.58  642.23
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -94.43796   17.73469  -5.325 1.02e-07 ***
## season        19.80282    1.81989  10.881  < 2e-16 ***
## yr            80.87881    2.16805  37.305  < 2e-16 ***
```

```
## mnth           0.04402    0.56749   0.078  0.93818
## hr             7.66437    0.16514  46.413  < 2e-16 ***
## holiday      -22.50906    7.88744  -2.854  0.00433 **
## weekday        0.41569    0.89519   0.464  0.64239
## workingday     5.12019    3.90854   1.310  0.19021
## weathersit    -3.09127    1.90792  -1.620  0.10520
## temp          81.48049   37.00823   2.202  0.02770 *
## atemp        229.80532   41.57557   5.527 3.30e-08 ***
## hum         -198.13969    6.89796 -28.724  < 2e-16 ***
## windspeed     43.16002    9.67612   4.460 8.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 141.9 on 17331 degrees of freedom
## Multiple R-squared:  0.3883, Adjusted R-squared:  0.3879
## F-statistic: 916.7 on 12 and 17331 DF,  p-value: < 2.2e-16
```

```
car::vif(fit1)
```

```
##     season         yr       mnth         hr    holiday    weekday workingday
##   3.498481   1.011440   3.283273   1.122711   1.499252   2.761087   2.841747
## weathersit       temp      atemp        hum  windspeed
##   1.278240  43.700269  43.938536   1.524266   1.200464
```

VIF is the Variance Inflation Factor and can be used to detect the presence of multicollinearity.

$$VIF = \frac{1}{(1 - R^2)}$$

According to Zuur et al. 2010, a VIF > 10 shows multicollinearity. But also more restrictive values such as 3 or even 1 can be chosen. We start with 10. temp and atemp show a high multicolinearity. Therefore, the atemp, which represents the feeling temperature, will be removed from the original dataset and temp will be kept.

```
dt <- dt %>% select(everything(), -atemp)
```

6.5 Train - test split

Split the data in a trainset with 75% of the data and a testset with 25% of the data

```
index <- createDataPartition(dt$cnt, p = 0.75, list=F)
dt.train <- dt[index, ]
dt.test <- dt[-index, ]
```

6.6 SVM model

6.6.1 Kfold cross validation

Setting the variable "kfold" to the method cross-validation and number of folds to 5.

```
kfold <- trainControl(method = 'cv', number = 5)
```

6.6.2 Linear kernel

Using a linear kernel an a k-fold cross validation

```
set.seed(9876)
```

```
svm.lin.1 <- train(cnt ~ ., data = dt.train,
                   trControl = kfold, method='svmLinear2',
                   tuneGrid = data.frame(cost = c(1))
                   )
print(svm.lin.1)
```

```
saveRDS(svm.lin.1, "svm.lin.1.rds")
```

```
svm.lin.1 <-readRDS("svm.lin.1.rds")
```

6.6.2.1 Save resp. Load Model Let's compare it to the linear kernel with the SVM implementation in the e1071 package.

```
set.seed(9876)
svm.lin.2 <- svm(cnt~., data = dt.train, kernel = 'linear',
                 type = 'eps-regression',
                 degree = 1,
                 coef0 = 1,
                 cost = 1,
                 cross = 5)


rmse(dt.train$cnt, svm.lin.2$fitted)
summary(svm.lin.2)
```

```
saveRDS(svm.lin.2, "svm.lin.2.rds")
```

```
svm.lin.2 <-readRDS("svm.lin.2.rds")
```

6.6.2.2 Save resp. Load Model The svm.lin.1 perform better and has a lower cost parameter. svm.lin.1 RMSE: 104.8963 and a cost of 1, compared to svm.lin.2 with a RMSE of 104.3788 and a cost of 1. The smaller the cost parameter, the more general the data is explained. If the cost parameter is chosen higher, the more specific it explains this data set. Therefore it is best to chose it as low as possible, with the best RSME value.If the cost value is higher, the model tend to overfit. The svm.lin.2 model has a slightely better RMSE and is much faster than the svm.lin.1. Therefore the svm.lin.2 is used for the prediction.

6.6.3 Prediction with linear kernel

```
set.seed(9876)
pred.linear <- predict(svm.lin.2, newdata=dt.test)
rmse(dt.test$cnt, pred.linear)
```

```
## [1] 106.1847
```

The linear kernel achieve a RMSE of 106.9027 for the prediction. Lets see if we can get better results with different kernels.

6.6.4 Polynomial kernel

We use the e1071 package to train a polynomial kernel. To tune the model, the degree, scale and C parameter can be changed to tune the model.

```
set.seed(9876)
#Polynomial: (gamma*u'*v + coef0)^degree
svm.poly.2 <- svm(cnt ~., data = dt.train, kernel = 'polynomial',
                  cross = 5, coef0 = 1, C = c(0.1, 0.25, 0.5, 1),
                  degree = 1)

svm.poly.2$coef0
svm.poly.2
rmse(dt.train$cnt, svm.poly.2$fitted)
svm.poly.2$degree
```

```
saveRDS(svm.poly.2, "svm.poly.2.rds")
```

```
svm.poly.2 <-readRDS("svm.poly.2.rds")
```

6.6.4.1 Save resp. Load Model The svm.poly.2 achieved a RMSE of 105.9098 with cost 1, degree 1 and coef 1. For the model testing, the svm.poly.2 will be used. Let's see how well the model works with the test data.

6.6.5 Prediction with polynomaial kernel

```
set.seed(9876)
pred.poly <- predict(svm.poly.2, newdata = dt.test)
rmse(dt.test$cnt, pred.poly)
```

```
## [1] 107.1937
```

The RMSE for the predictions on the testdata for the svm with a polynomial kernel is 108.1839. Let's have a look how a svm with a radial kernel performec.

6.6.6 Radial kernel

```
set.seed(9876)

svm.rad.1 <- train(cnt ~., data=dt.train,
                   trConrol = kfold,
                   method = 'svmRadial')
summary(svm.rad.1)
print(svm.rad.1)
```

```
saveRDS(svm.rad.1, "svm.rad.1.rds")
```

```
svm.rad.1 <-readRDS("svm.rad.1.rds")
```

6.6.6.1 Save resp. Load Model The svmRadial kernel uses the kernlab package. The radial kernel performed so far best, with a RSME of 51.06311. The final values used for the model were sigma = 0.0107185 and C = 1.

6.6.7 Prediction with radial kernel

```
set.seed(9876)
pred.radial <- predict(svm.rad.1, newdata = dt.test)
rmse(dt.test$cnt, pred.radial)
```

```
## [1] 45.76372
```

The RMSE for the predicted values with the SVM with a radial kernel is 49.56235.

6.6.8 Model comparision

Comparing the predictions based on their RMSE

```
set.seed(9876)
#linear model
rmse(dt.test$cnt, pred.linear)
```

```
## [1] 106.1847
```

```
#polynomial model
rmse(dt.test$cnt, pred.poly)
```

```
## [1] 107.1937
```

```
#radial model
rmse(dt.test$cnt, pred.radial)
```

```
## [1] 45.76372
```

The radial kernel clearly outperformed the other two. the radial kernel has a RSME of 49.55235, the polynomial an RSME of 108.1839 and the linear kernel a RSME of 106.9027. Therefor the winner of the support vector machine models is the one with the radial kernel. It also takes the most time to run. The linear model is much faster but has a higher RMSE value.

7 Artificial Neural Network

7.1 Data Preparation

7.1.1 Load data

Before starting to create NN models, all categorical variables are converted into a "one-hot" resp. binary variable. This is told to be best practice, as it should enhance the prediction performance let the model computing converge faster. Latter is also of great interest due to the long calculation time of NN models. Also, the dependent variable is normalized.

7.1.2 Break multi factor variables down into single factor variables

df$season_1$ < $-ifelse(df$season == "1", 1, 0) df$season_2$ < $-ifelse(df$season == "2", 1, 0) df$season_3$ < $-ifelse(df$season == "3", 1, 0) ...

7.1.3 Normalize dependent variable

```
data <- df
data$cnt <- (df$cnt - min(df$cnt)) / (max(df$cnt) - min(df$cnt))
```

7.2 ANN Model 1: Computing a RAnge of Model

s a first approach to the data set, a few models are computed at once with a preset range of layer. A slightly reduced training data volume is taken (65%). Each model is five-times cross validated. Also, the threshold is set to 0.1, which is high error tolerance value. Those first parameter settings are chosen in order to compute quantity rather than quality. The result shown below suggests that there is no need of a third nor a second layer, and that the first layer doesn't improve with more than four neurons in the first layer. The Root Mean Square Error is about 183. A prediction-vs-real data plot provides a visualization of the model performance. This plot suggests that there might be potential for improvement.

The output neuron is set with "linear.output = TRUE", which means that the output is a continuous number and not a factor.

7.2.1 Split Data into Train and Test Partition

```
set.seed(123)
indices_1 <- createDataPartition(data$cnt, p=.65, list = F)
train_1 <- data %>% slice(indices_1)
test_1 <- data %>% slice(-indices_1)
```

7.2.2 Set up Parameter Range

```
set.seed(44)
tuGrid_1 <- expand.grid(.layer1=c(1,2,4:8), .layer2=c(0,2,3,4), .layer3=c(0,2))
trCtrl_1 <- trainControl(
  method = 'repeatedcv',
  number = 5,
  repeats = 1,
  returnResamp = 'final')
```

7.2.3 Train Models

```
models_1 <- train(cnt ~ hum + temp + weathersit_1 + weathersit_2 + weathersit_3 + hr_1 + hr_2 + hr_3 + h
                  method = 'neuralnet',
                  metric = 'RMSE',
                  linear.output = TRUE,
                  threshold = 0.1,
                  lifesign.step = 1000,
                  lifesign = "full",
                  preProcess = c('center', 'scale'),
                  tuneGrid = tuGrid_1,
                  trControl = trCtrl_1)
```

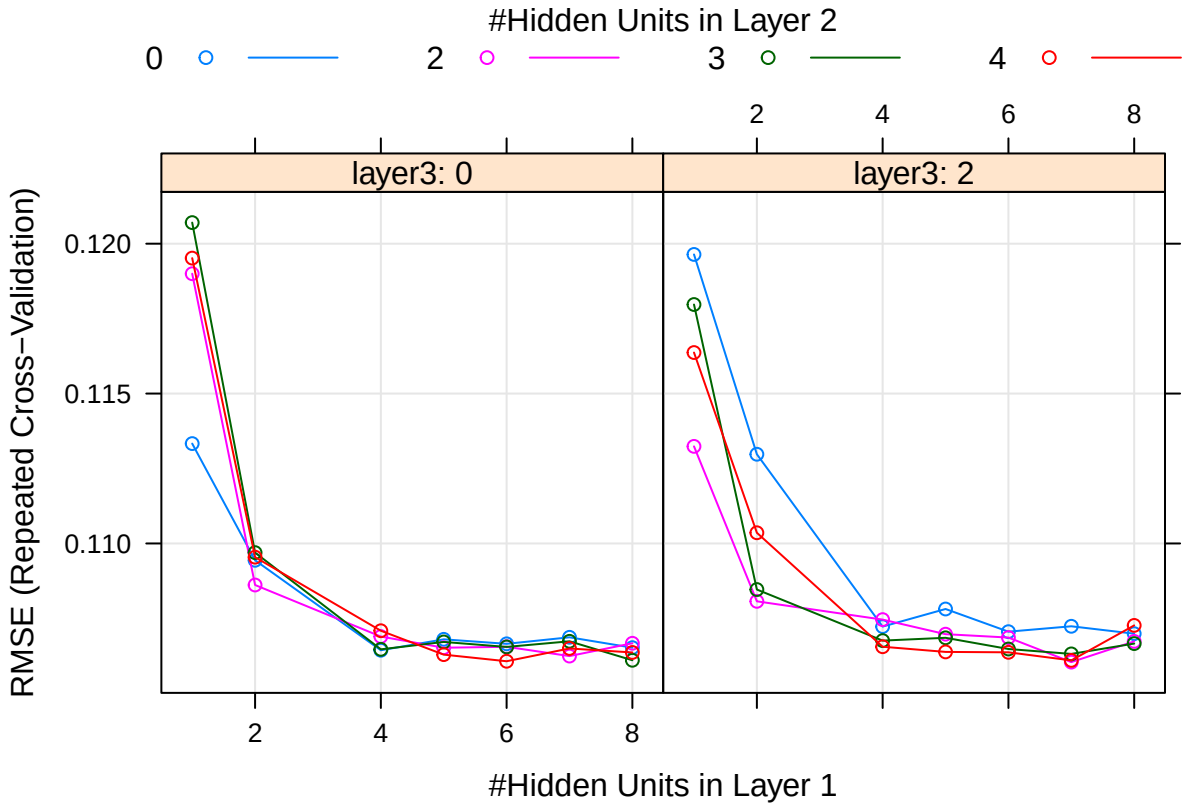
7.2.4 Save resp. Load Model

```
saveRDS(models_1, "neural_nets_models_1.rds")
```

```
models_1 <-readRDS("neural_nets_models_1.rds")
```

7.2.5 Plot Models

```
plot(models_1)
```



7.2.6 Compute Prediction with best Model

```
pred_1 <- compute(models_1$finalModel, test_1 %>% select(-cnt))
pred_1 <- pred_1$net.result * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
control_1 <- test_1$cnt * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
```

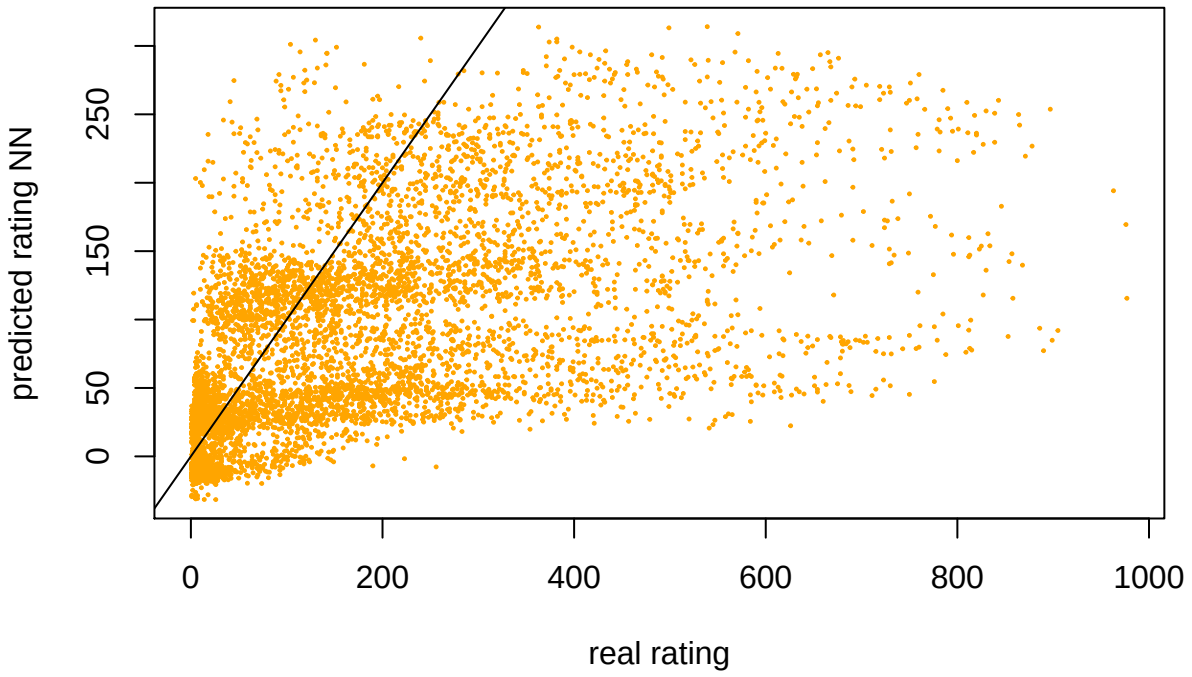
7.2.7 Root Mean Square

```
sqrt(mean((control_1 - pred_1)^2))
```

```
## [1] 183.5873
```

7.2.8 Plot Prediction against Real Data

```
plot(control_1, pred_1, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real rati
abline(0,1)
```



7.3 ANN Model 2: Layer 4/0/0, Threshold 0.01

After the first investigation in the NN layer's behavior, the four-neurons model layout is taken to be computed again. This time, a higher data amount (80%) and an error threshold of 0.01 is set. The result of this more meticulous model shows already great improvement in RSME and the plot is also more satisfying. It is unclear if a model with higher complexity would have performed even better with a 0.01 threshold, but this is the trade-off for a much faster computing time.

7.3.1 Split Data into Train and Test Partition

```
set.seed(42)
indices_2 <- createDataPartition(data$cnt, p=.8, list = F)

train_2 <- data %>% slice(indices_2)
test_2 <- data %>% slice(-indices_2)
```

7.3.2 Train Model

```
set.seed(42)
model_2 = neuralnet(cnt ~ hum + temp + weathersit_1 + weathersit_2 + weathersit_3 + hr_1 + hr_2 + hr_3 +
```

7.3.3 Save resp. Load Model

```
saveRDS(model_2, "neural_nets_model_2.rds")
```

```
model_2 <-readRDS("neural_nets_model_2.rds")
```

7.3.4 Compute Prediction

```
pred_2 <- compute(model_2, test_2 %>% select(-cnt))
pred_2 <- pred_2$net.result * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
```

```
control_2 <- test_2$cnt * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
```

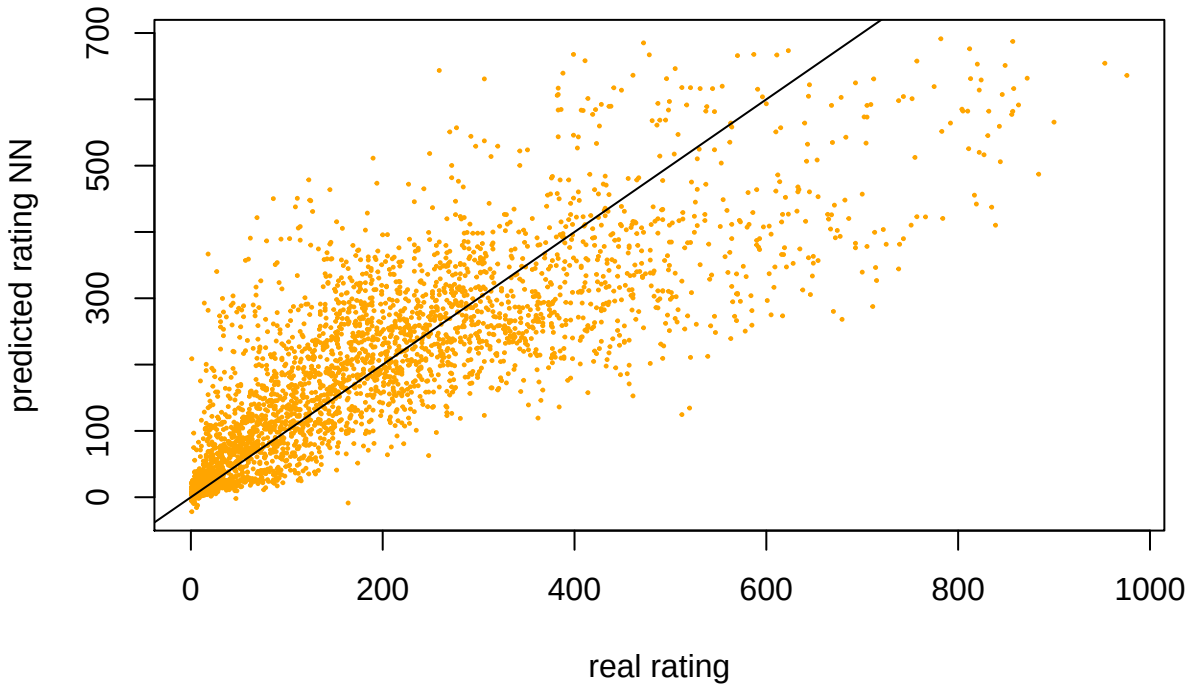
7.3.5 Root Mean Square

```
sqrt(mean((control_2 - pred_2)^2))
```

```
## [1] 100.7022
```

7.3.6 Plot Prediction against Real Data

```
plot(control_2, pred_2, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real rati
abline(0,1)
```



7.4 ANN Model 4: more Predictors, Layer 4/0/0, Threshold 0.05

Now more predictors are added to the model; the weekdays, the months and the wind speed. Again with a four-neuron model, the model is computed with a threshold of 0.05. The results shows improvement in RSME (77) and thus suggest that there is valuable information in the added predictors. Again, the plot appear to be better.

7.4.1 Train Model

```
set.seed(41)
model_4 = neuralnet(cnt ~ hum + temp + windspeed + weathersit_1 + weathersit_2 + weathersit_3 + hr_1 + h
```

7.4.2 Load model (computed in advance)

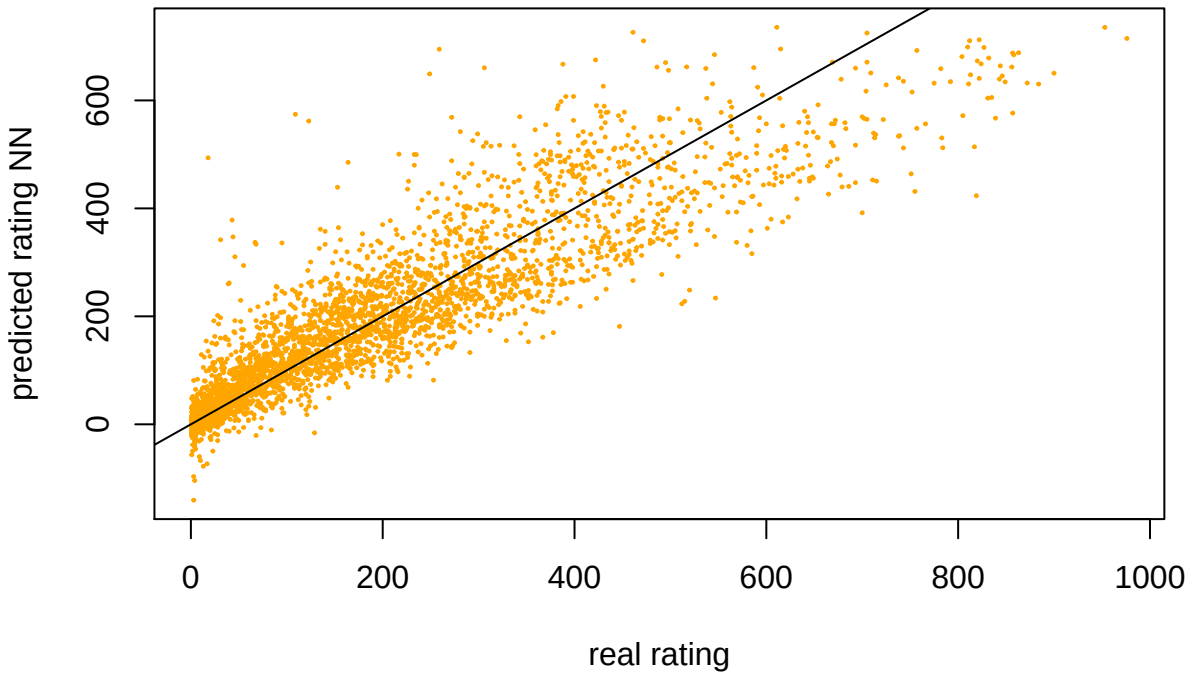
```
model_4 <-readRDS("neural_nets_model_4.rds")
```

7.4.3 Compute Prediction

7.4.4 Root Mean Square

```
## [1] 77.30468
```

7.4.5 Plot Model Performance



7.5 ANN Models 5: Computing a Range of Model

With the new predictors added, the model might benefit from a new layer architecture. Again, a range of model is computed. But this time, more data (80%) and new predictors are added. The result suggest a model architecture of 7,3,0 neurons.

7.5.1 Set up Parameter Range

```
set.seed(44)
tuGrid_2 <- expand.grid(.layer1=c(4:8), .layer2=c(0,2,3,4), .layer3=c(0,2))

trCtrl_2 <- trainControl(
  method = 'repeatedcv',
  number = 5,
  repeats = 1,
  returnResamp = 'final',
)
```

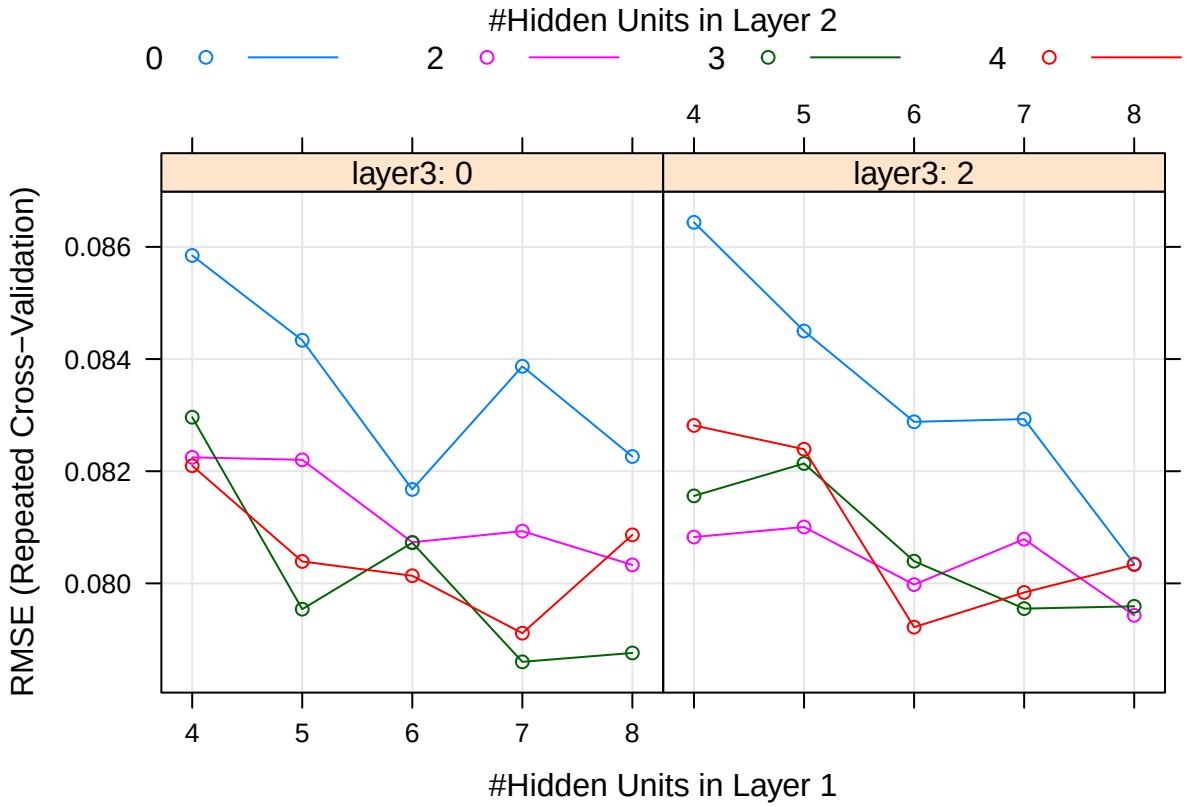
7.5.2 Train Models

```
models_5 <- train(cnt ~ hum + temp + windspeed + weathersit_1 + weathersit_2 + weathersit_3 + hr_1 + hr
  method = 'neuralnet',
  metric = 'RMSE',
  linear.output = TRUE,
  threshold = 0.1,
```

```
  stepmax = 250000,
  lifesign.step = 1000,
  lifesign = "full",
  preProcess = c('center', 'scale'),
  tuneGrid = tuGrid_2,
  trControl = trCtrl_2)
```

7.5.3 Plot Models

```
plot(models_5)
```



7.5.4 Compute Prediction

```
pred_5 <- compute(models_5$finalModel, test_2 %>% select(-cnt))
pred_5 <- pred_5$net.result * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
control_5 <- test_2$cnt * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
```

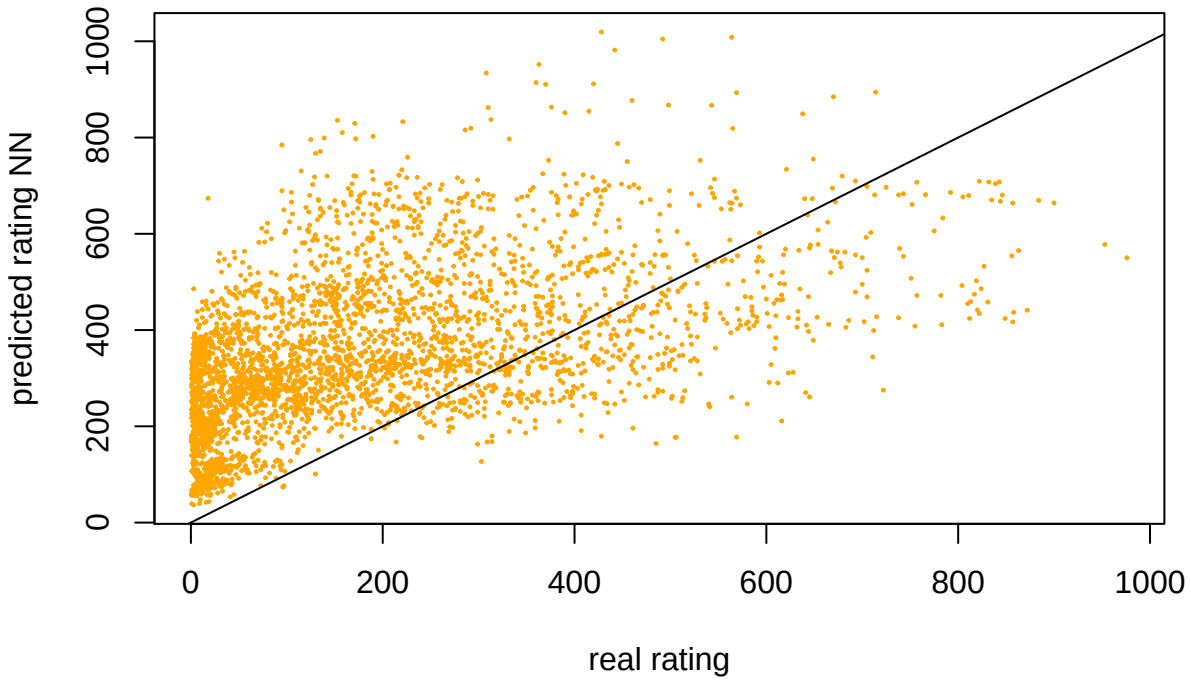
7.5.5 Root Mean Square

```
sqrt(mean((control_5 - pred_5)^2))
```

```
## [1] 238.1291
```

7.5.6 Plot Prediction against Real Data

```
plot(control_5, pred_5, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real rati
abline(0,1)
```



##

ANN Model 6: Layer 7/3/0, Threshold 0.01 The 7,3,0 model is computed again but with higher "resolution". The RSME drops down to 70.

7.5.7 Compute Prediction

```
pred_6 <- compute(model_6, test_2 %>% select(-cnt))
pred_6 <- pred_6$net.result * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
control_6 <- test_2$cnt * (max(df$cnt) - min(df$cnt)) + min(df$cnt)
```

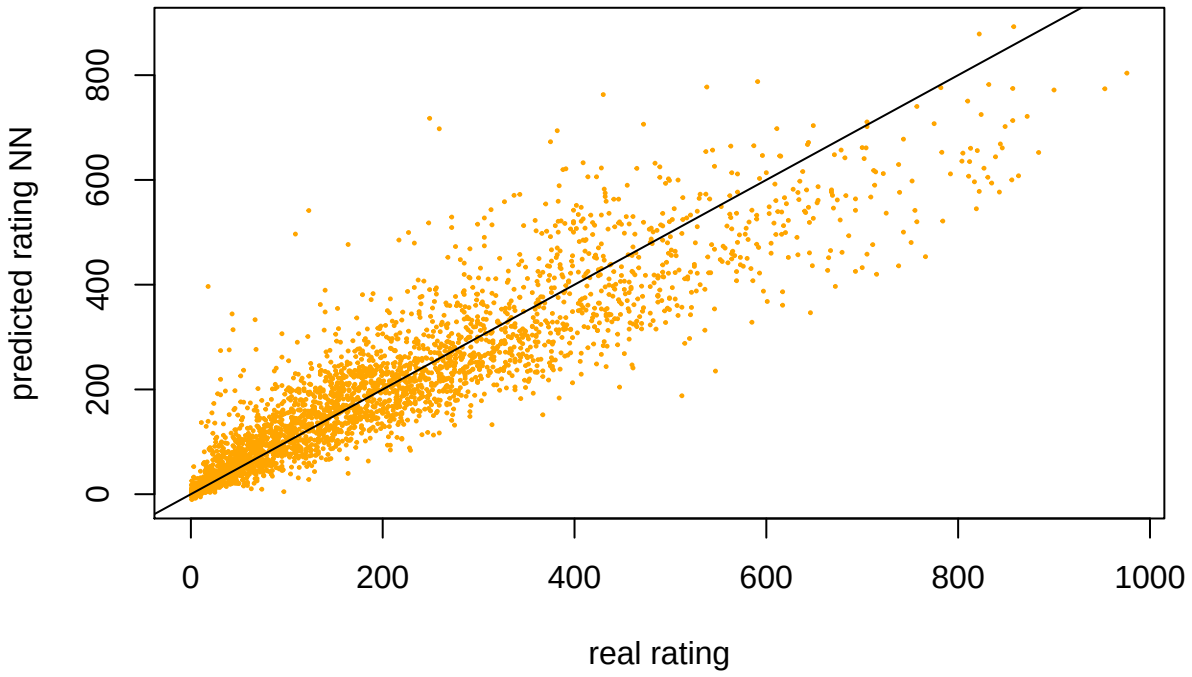
7.5.8 Root Mean Square

```
sqrt(mean((control_6 - pred_6)^2))
```

```
## [1] 70.65027
```

7.5.9 Plot Prediction against Real Data

```
plot(control_6, pred_6, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real rati
abline(0,1)
```



7.6 ANN Model 9: New Pred.: Year, Layer 7/3/0, Threshold 0.01

After some reflections about the data, one potential mistake stood out. Initially, the variable "year" was left out, since the data set covers only two years. But, as there is an increasing trend in the bike rental over the two years, there might be information in this predictor. The predictor "year" is added to the model 7,3,0. The result shows a great improvement in RSME with a drop to 49. The plot shows an overall better prediction behavior.

```
model_9 <-readRDS("neural_nets_model_9.rds")
```

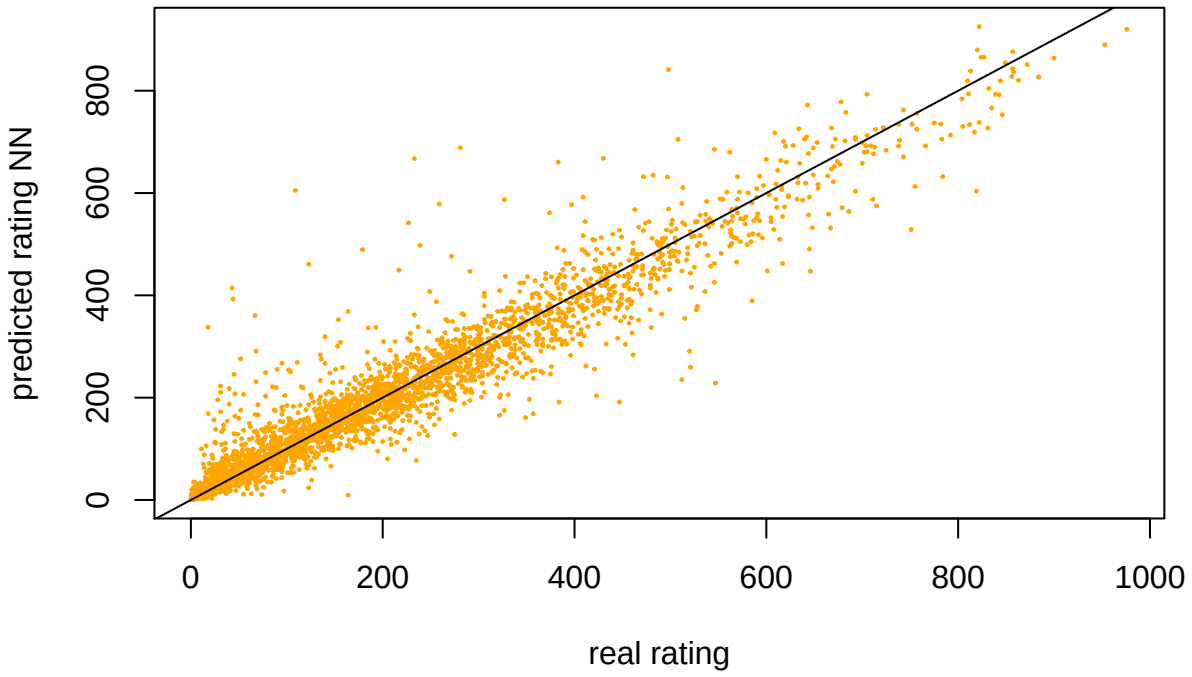
7.6.1 Compute Prediction

7.6.2 Root Mean Square

```
## [1] 49.18283
```

7.6.3 Plot Prediction against Real Data

```
plot(control_9, pred_9, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real rati
abline(0,1)
```



7.7 ANN Models 10: Computing a RAnge of Model

7.7.1 Set up Parameter Range

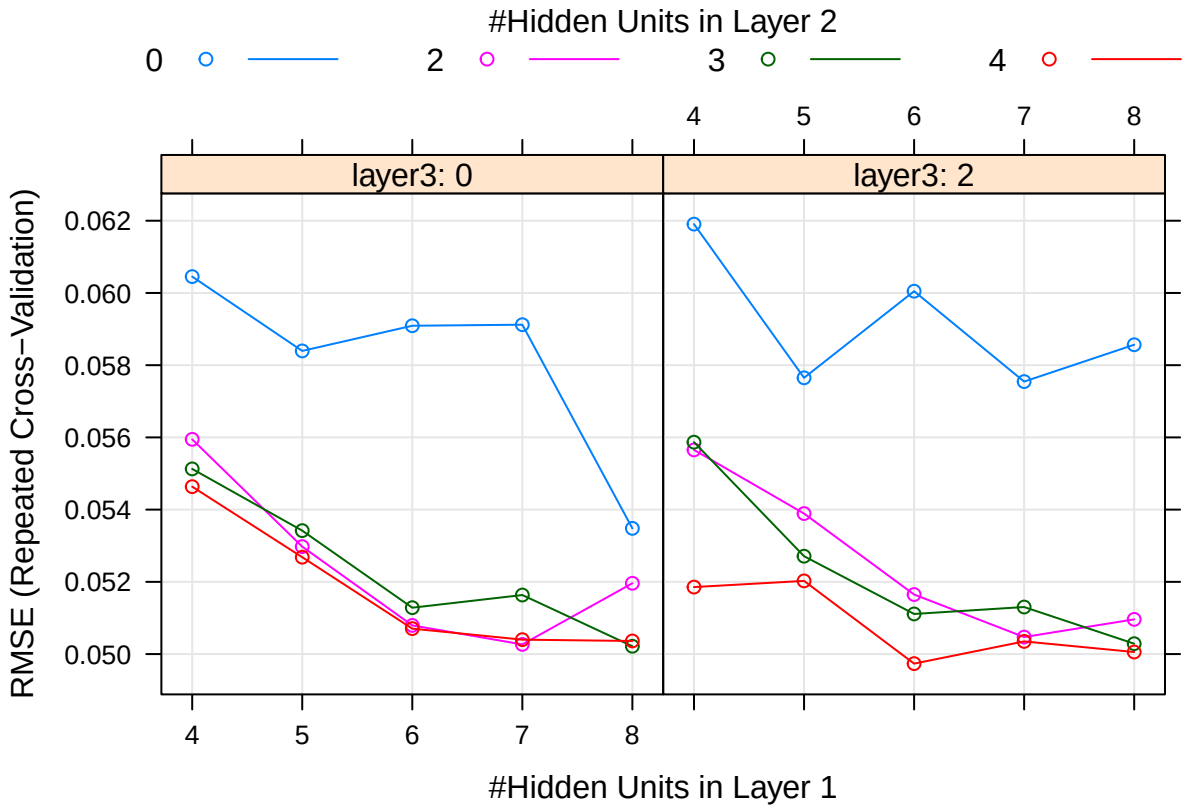
```
set.seed(41)
tuGrid_3 <- expand.grid(.layer1=c(6, 8, 10, 12, 14), .layer2=c(6, 8, 10), .layer3=c(2, 3, 4))

trCtrl_3 <- trainControl(
  method = 'repeatedcv',
  number = 3,
  repeats = 1,
  returnResamp = 'final',
)
```

7.7.2 Train Models

```
models_10c <- train(cnt ~ workingday + holiday + yr + hum + temp + windspeed + weathersit_1 + weathersi
  method = 'neuralnet',
  metric = 'RMSE',
  linear.output = TRUE,
  threshold = 0.05,
  stepmax = 450000,
  lifesign.step = 1000,
  lifesign = "full",
  preProcess = c('center', 'scale'),
  tuneGrid = tuGrid_3,
  trControl = trCtrl_3
  )
```

```
models_10 <-readRDS("neural_nets_models_10.rds")
```

```
plot(models_10)
```



7.8 ANN Model 11

After progressively adding the predictors to the model, the conclusion is that nearly all variables in the data set are useful as predictor, the remaining last two variables; "working day" and "holiday" are finally added to the model. Those two variables might help the model to understand the the case of rarer events, such as holiday or public holiday. This two predictors slightly improved the model to an RSME of 46.

```
model_11 <-readRDS("neural_nets_model_11.rds")
```

7.8.1 Compute Prediction

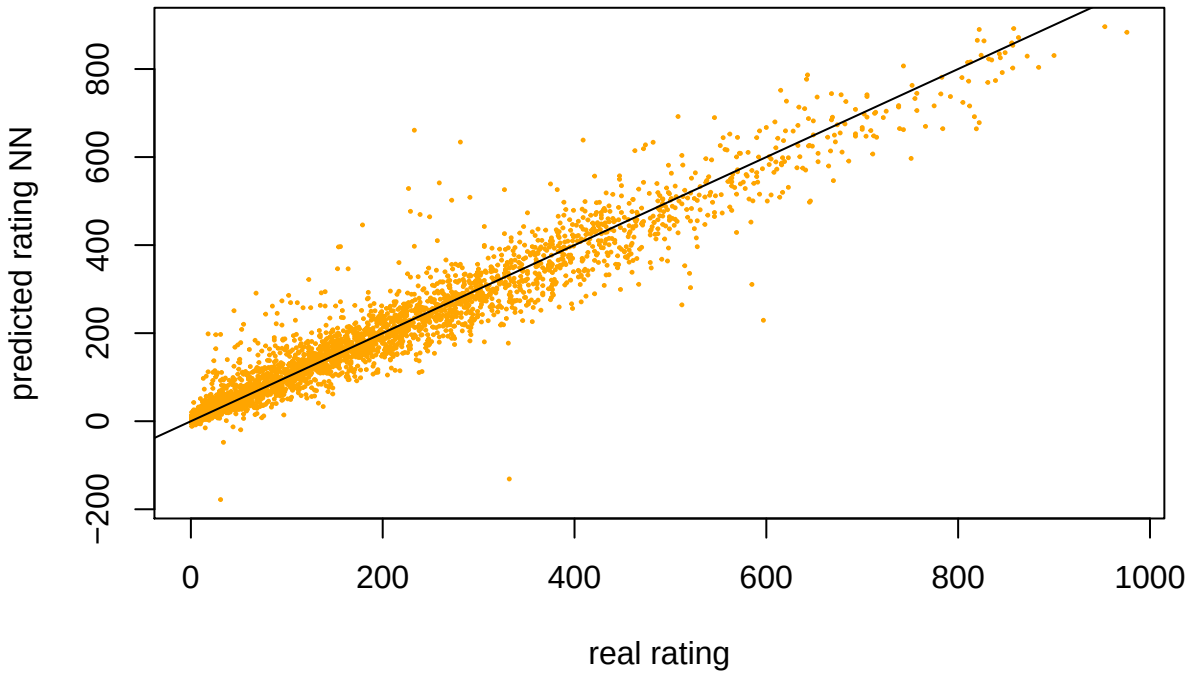
7.8.2 Root Mean Square

```
sqrt(mean((control_11 - pred_11)^2))
```

```
## [1] 46.02274
```

7.8.3 Plot Prediction against Real Data

```
plot(control_11, pred_11, col='orange', cex = .3, pch=20, ylab = "predicted rating NN", xlab = "real ra
abline(0,1)
```



7.9 Further models

The best prediction performance is achieved with a 14/8/2 layer architecture, with a RSME of 45. Further attempts with much higher layer architecture, for example the approach with 2/3 of the predictors as input neurons -> 34/12/4 do not perform better in RSME (54).

7.10 ANN Reflection & Conclsion:

- Being able to save models with saveRDS() & readRDS() is a great relief when working with NN.
- It seems that R studio doesn't use the full computational potential of the CPU. It turned out that already an Intel quad core 8th gen mobile is able to process two model simultaneously, without any "noticeable" loss in computing speed. This come especially handy as training more elaborated NN models becomes heavily time consuming.
- Model with higher amount of neurons generally converged in fewer iteration steps. But in the end, as each iteration takes longer to be computed, the model takes more time to converged.
- One should adapt faster the number of neurons when increasing predictors.

8 Optimization Problem

For our use case we created the following hypothetical optimization problem:

At this moment we own 2'000 bikes. As we want to grow and get the most out of the demand, we plan on buying more bikes. This means we want to be able to cover as much of the demand as possible without having too much downtime (bikes that we own, but are not used by customers).

Furthermore, we want the investment into the new bikes being paid off within 2 years.

To buy a new bike will cost us 300.$The bikes we own cost us on average$1 a day for maintenance and so on, regardless whether they are in use or not. If a bike is in use, we will earn 3$ a day.

The demand depends on the months. The daily demand coefficients look as follows:

```
Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
```

a: 1.5 1.5 1.9 2.3 2.7 3.0 3.2 3.5 3.5 3.3 3.0 2.2 a1 a2 ... a12

They are used with the factor 1'000. So demand for June would be 3.0 * 1'000 = 3'000 bikes per day

Furthermore the months have a different numbers of days.

```
  Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
```

b: 31 28 31 30 31 30 31 31 30 31 30 31 b1 b2 ... b12

Therefor the maximum revenue per day is the smaller number of either the maximum demand or the bikes we own that day.

In mathematical terms we get the following equation:

$$x * 300 = 2 * (3 * \sum_{i=1}^{12} (b_i * min(x + 2000, a_i * 1000) - 1 * 365 * (x + 2000)))$$

Respectively resolved to 0:

$$0 = 2 * (3 * \sum_{i=1}^{12} (b_i * min(x + 2000, a_i * 1000) - 1 * 365 * (x + 2000))) - x * 300$$

This equation resolved to x will give es the maximum amount of bikes we should buy, if we want the investment to be paid off in two years.

#Conclusion ## The best Models based on the RMSE values

```
#RMSE Linear Model RMSE of 97.4
RMSE_Linear_Model <- sqrt(mean((exp(control_log) - exp(predict_3))^2))

#RMSE Gam Model (RMSE of 90.4)
RMSE_gam_5 <- sqrt(mean((exp(control_log) - exp(predict_gam_5))^2))

#RMSE GLM
# Binomial (RMSE of 0.06111774)
RMSE_glm_binomial_3 <- rmse((day.test_in$registered/day.test_truth), predicted.binomial.test.3)
# Poisson (RMSE of 134 for hourly data)
RMSE_glm_poisson_1 <- rmse(day.test_truth, predicted.poisson.test.1)
```

```
## Warning in actual - predicted: longer object length is not a multiple of shorter
## object length
```

```
#RMSE SVM Model wit a radial kernel (RSME o 47.66508)
RMSE_svm <- rmse(dt.test$cnt, pred.radial)

#RMSE NN Model
45
```

```
## [1] 45
```

8.1 Which model should the company use?

Justification, best model to find best maintenance window