Machine learning 2

Mara Acuja

24 11 2021

notes

```r
print("The paper talks abotu 73 different countries/areas, but when visualizing them there are fewer do
```

```
## [1] "The paper talks abotu 73 different countries/areas, but when visualizing them there are fewer do
```

```r
print("scatter plot für koordinaten und box plot für varianz in long und lat")
```

```
## [1] "scatter plot für koordinaten und box plot für varianz in long und lat"
```

```r
print("boxplot für distances für jedes modell -> um diese vergleichen zu können")
```

```
## [1] "boxplot für distances für jedes modell -> um diese vergleichen zu können"
```

```r
print("für finales ergebniss, ein karte, welche ist und soll koorinaten anzeigt und mit linien verbindet
```

```
## [1] "für finales ergebniss, ein karte, welche ist und soll koorinaten anzeigt und mit linien verbinde
```

```r
print("for pre-processing mabe a small whiskers plot, that proves that the data is already normalized")
```

```
## [1] "for pre-processing mabe a small whiskers plot, that proves that the data is already normalized"
```

librarys

```r
#install.packages("installr")
#library("installr")
#install.Rtools()

#install.packages("ggmap")
#install.packages("maptools")
#install.packages("maps")
#install.packages("glmnet")
#install.packages("ISLR")
#TODO the one below necessary?
#install.packages("rgl")

library("glmnet")
```

```
## Lade nötiges Paket: Matrix
```

```
## Loaded glmnet 4.1-3
```

```r
library("ggplot2")
library("ggmap")
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library("maptools")
```

```
## Lade nötiges Paket: sp

## Checking rgeos availability: FALSE
## Please note that 'maptools' will be retired by the end of 2023,
## plan transition at your earliest convenience;
## some functionality will be moved to 'sp'.
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcli
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library("maps")

library(ISLR)
options(rgl.printRglwidget = TRUE)
library(rgl)
```

Comparison of machine learning algorithms / Introduction / Theory

This report attempts to compare two regression/classification algorithms on its behaviour on a specific data set. What should be found out? This report attempts to compare two regression/classification algorithms on its behavior on a specific data set. What should be found out?

Data set

Here should be a bit of a short summary of the data set with some key characteristics of the data set.

What is it about? Which variables are included? What type of variables? What is missing? What about missing values?

```
distances <- function(predicted, actual_value) {
  dif <- predicted-actual_value
  dif <- dif * (40030/360) # scaling coordinates to km by the factor circumference (km) / 360°
  mse <- sqrt(dif[,1]^2 + dif[,2]^2)
  return(mse)
}
```

```
data <- read.csv("Data/default_plus_chromatic_features_1059_tracks.txt", header=FALSE)
data <- as.data.frame(data)
colnames(data)[117:118] <- c("Latitude", "Longitude")
```

```
# Maybe some more preprocessing could be done here.
anyNA(data) # testing if there is at least a single NA -> but in this dataset there isn't
```

```
## [1] FALSE
```

```
# Maybe some more pre-processing could be done here.
anyDuplicated(data) # testing for duplicates -> 0 found
```

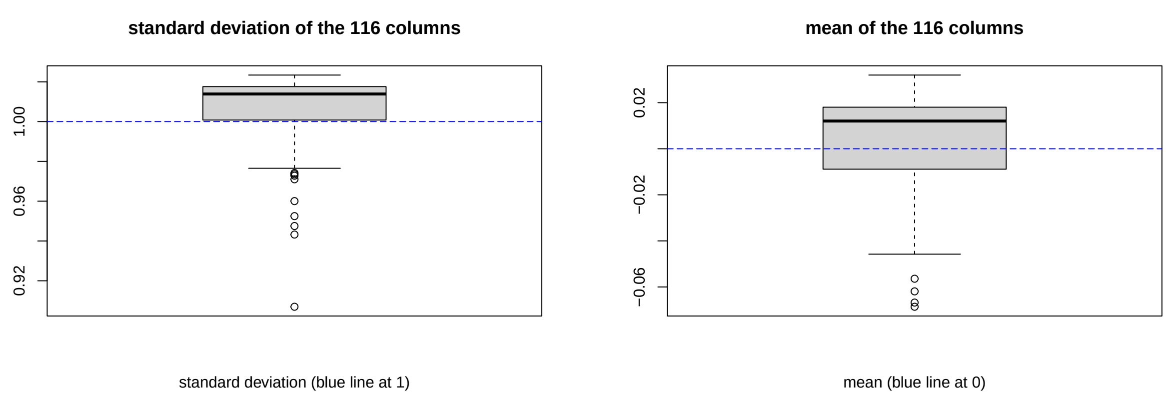
```
## [1] 0
```

```
# feature scaling
```

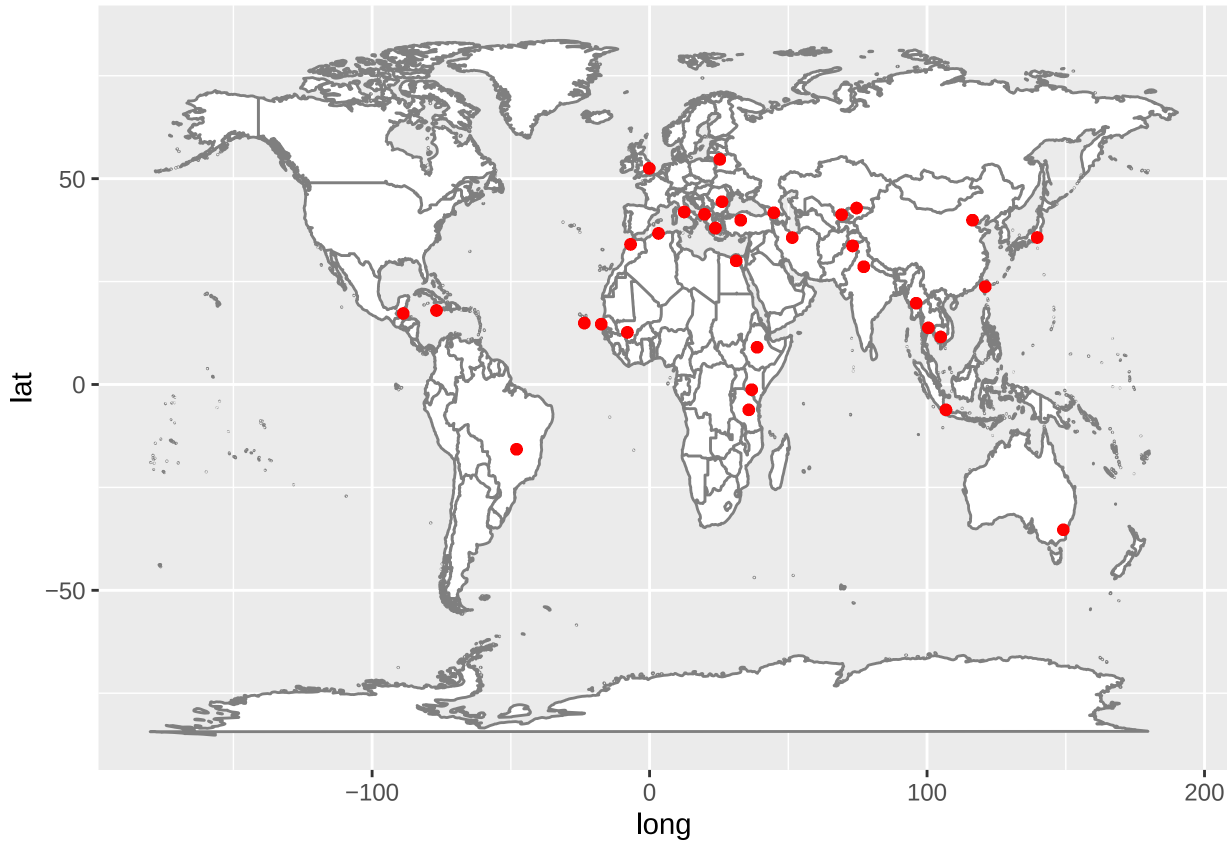
```
# separate labels from features
#label_column_names <- c("Longitude", "Latitude")
#data_labels <- data[label_column_names]
#data_features <- subset(data, select = -label_column_names)
# scale features
#data_features_scaled <- as.data.frame(scale(data_features))
# add labels to the now scaled features
#data <- cbind(data_features_scaled, data_labels)
```

some insights into the data

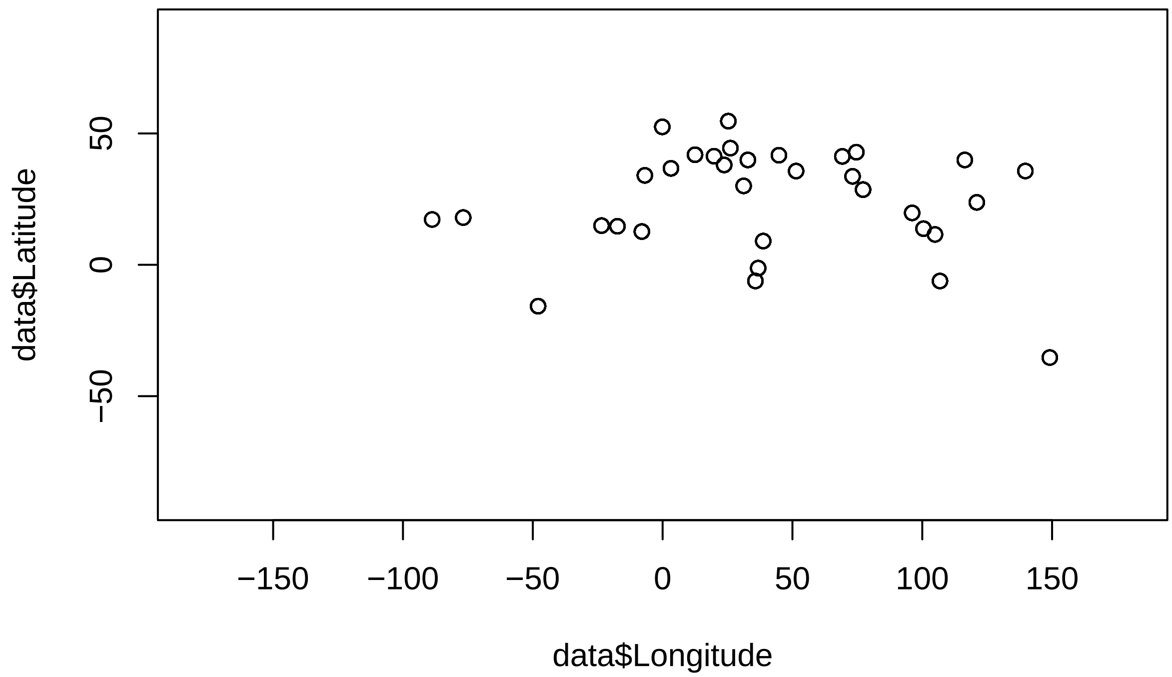
```
# check if data is already standardized

# get columns without the target cols ("long" and "lat")
data_without_target_cols <- subset(data, select=-c(Latitude,Longitude))

# for each column in the data get sd and median
sd_per_col <- apply(data_without_target_cols, 2, sd) # the two stands for columns, if we would have use
sd_per_col_df <- data.frame(sd_per_col)

mean_per_col <- apply(data_without_target_cols, 2, mean)
mean_per_col_df <- data.frame(mean_per_col)

sd_and_mean_per_col_df <- merge(sd_per_col_df, mean_per_col_df, by="row.names", all=TRUE)

#par(mar = c(4, 4, .1, .1)) # to make the two plots show side by side and not above each other

boxplot(sd_per_col, data=sd_and_mean_per_col_df, xlab="standard deviation (blue line at 1)", y_lab="valu
abline(h=1, col = "blue", lty=5)

boxplot(mean_per_col, data=sd_and_mean_per_col_df, xlab="mean (blue line at 0)", y_lab="value", main="me
abline(h=0, col = "blue", lty=5)
```



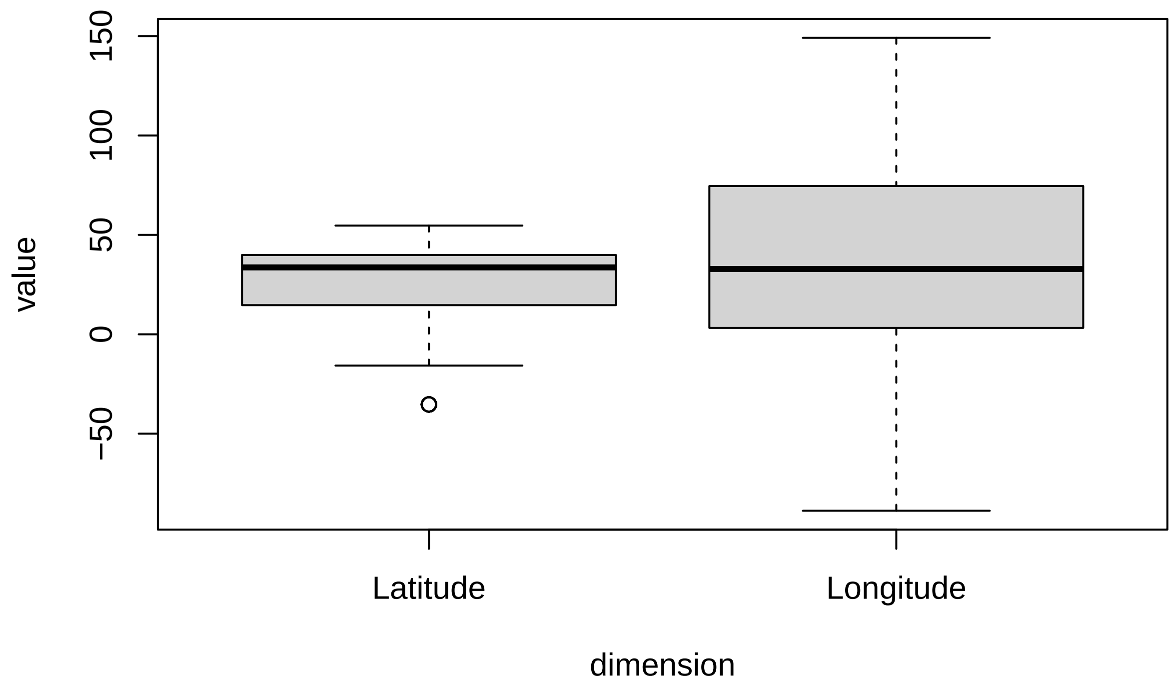

```
# basic world map with music origins
mapWorld <- borders("world", colour="gray50", fill="white")
mp <- ggplot() + mapWorld

mp + geom_point(data = data, aes(x = Longitude, y = Latitude), color = "red", alpha = 0.5)
```




```r
plot(data$Longitude, data$Latitude, xlim=c(-180, 180), ylim=c(-90, 90))
```




```r
df_lat <- data.frame(value = data$Latitude, dimension = "Latitude")
df_long <- data.frame(value = data$Longitude, dimension = "Longitude")
boxplot_df <- rbind(df_lat, df_long)

boxplot(value~dimension, data=boxplot_df)
```




```
set.seed(1)
n <-dim(data)[1]
train <- sample(1:n, 0.8*n)
test <- (1:n)[-train]
```

Method

Short summary about the algorithms. Which are used? What do we do? Classification or Regression?

Baseline - Linear Regression

First, we take a baseline to get a basic understanding on how well our chosen algorithms perform. Therefore, we decided to use a linear regression. First we created a model which includes all variables. The lm() command cannot compute a model for both output variables at the same time. So it creates two separate linear models, one for each output variable:

```
model.lm.all <- lm(cbind(Longitude, Latitude)~., data=data[train,])
pred.all <- predict(model.lm.all, newdata=data[test,])
```

To calculate a good meaningful measurement for the goodness of fit for the predictions the distance from the true location is calculated. The distances are calculated as the euclidean distances of the Longitude and Latitude between the predictions and the true location. They are measured in [km]. These will be used for all the comparisons of the algorithms between each other but also with the baseline and also with the literature (we need here another citation to the original paper)

The predictions are on average quite far from the true destination:

```
## [1] 5850.727
```

The impact of the variables was analysed. It seemed that not all of them have a significant influence on the models. So a second model was developed, just using the variable which have a significiant influence on the full model, either on the Latitude or on the Longitude.

```r
model.lm.sig <- lm(cbind(Longitude, Latitude)~ V4+V9+V16+V30+V32+V33+V37+V38+V61+V90+V91+V92+V95+V96
                   +V104+V5+V6+V8+V9+V11+V15+V34+V39+V63+V94+V97,
                   data=data[train,])
pred.sig <- predict(model.lm.sig, newdata=data[test,])
```

The predictions for the smaller models are on average a bit better:

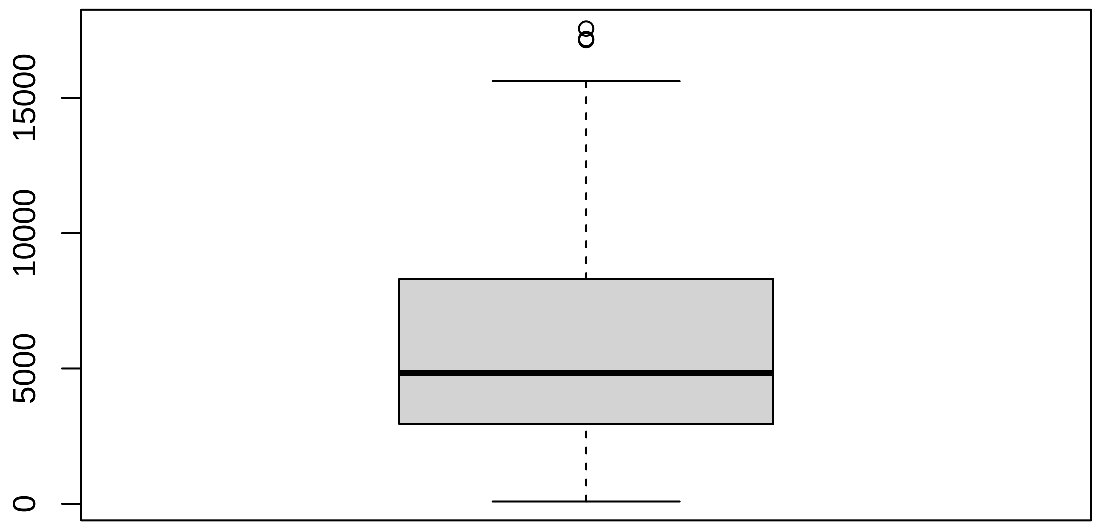
```
## [1] 5635.057
```

An ANOVA was calculated to check if there is a significant difference between the two models.

```r
anova(model.lm.all, model.lm.sig)
```

```
## Analysis of Variance Table
##
## Model 1: cbind(Longitude, Latitude) ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
##     V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 +
##     V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 +
##     V28 + V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 +
##     V38 + V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 +
##     V48 + V49 + V50 + V51 + V52 + V53 + V54 + V55 + V56 + V57 +
##     V58 + V59 + V60 + V61 + V62 + V63 + V64 + V65 + V66 + V67 +
##     V68 + V69 + V70 + V71 + V72 + V73 + V74 + V75 + V76 + V77 +
##     V78 + V79 + V80 + V81 + V82 + V83 + V84 + V85 + V86 + V87 +
##     V88 + V89 + V90 + V91 + V92 + V93 + V94 + V95 + V96 + V97 +
##     V98 + V99 + V100 + V101 + V102 + V103 + V104 + V105 + V106 +
##     V107 + V108 + V109 + V110 + V111 + V112 + V113 + V114 + V115 +
##     V116
## Model 2: cbind(Longitude, Latitude) ~ V4 + V9 + V16 + V30 + V32 + V33 +
##     V37 + V38 + V61 + V90 + V91 + V92 + V95 + V96 + V104 + V5 +
##     V6 + V8 + V9 + V11 + V15 + V34 + V39 + V63 + V94 + V97
##   Res.Df Df Gen.var.  Pillai approx F num Df den Df    Pr(>F)
## 1    774      666.34
## 2    821 47   688.29 0.17239   1.5533     94   1548 0.0007525 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model tells that there is a difference and, as seen before, the smaller model performs better.



Algorithm 1 - Ridge Regression

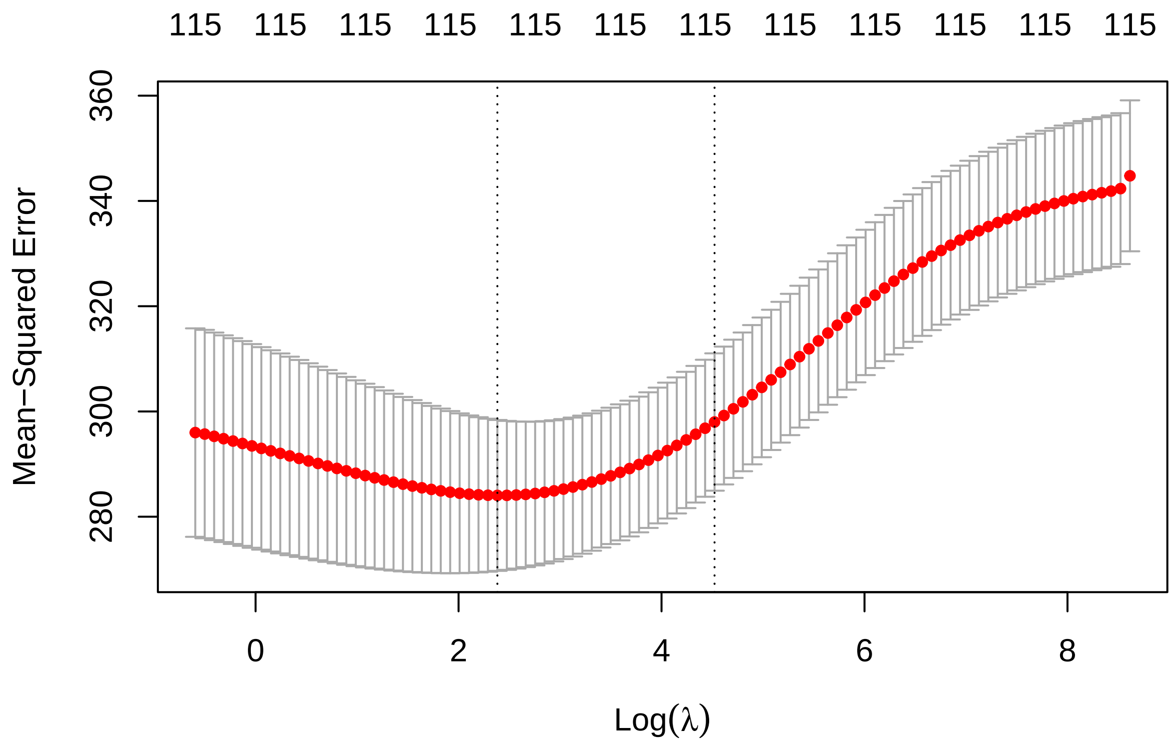
Short introduction of the first algorithm. What does it do? What are the strengths? What are weaknesses? How is it implemented, including major code snippets.

The first algorithm we tried was ridge regression. This algorithm is similar to a linear regression but while the linear regression tries to minimize the difference between the weighted input variables and the output data, the ridge regression adds a regularisation term on the input variables.

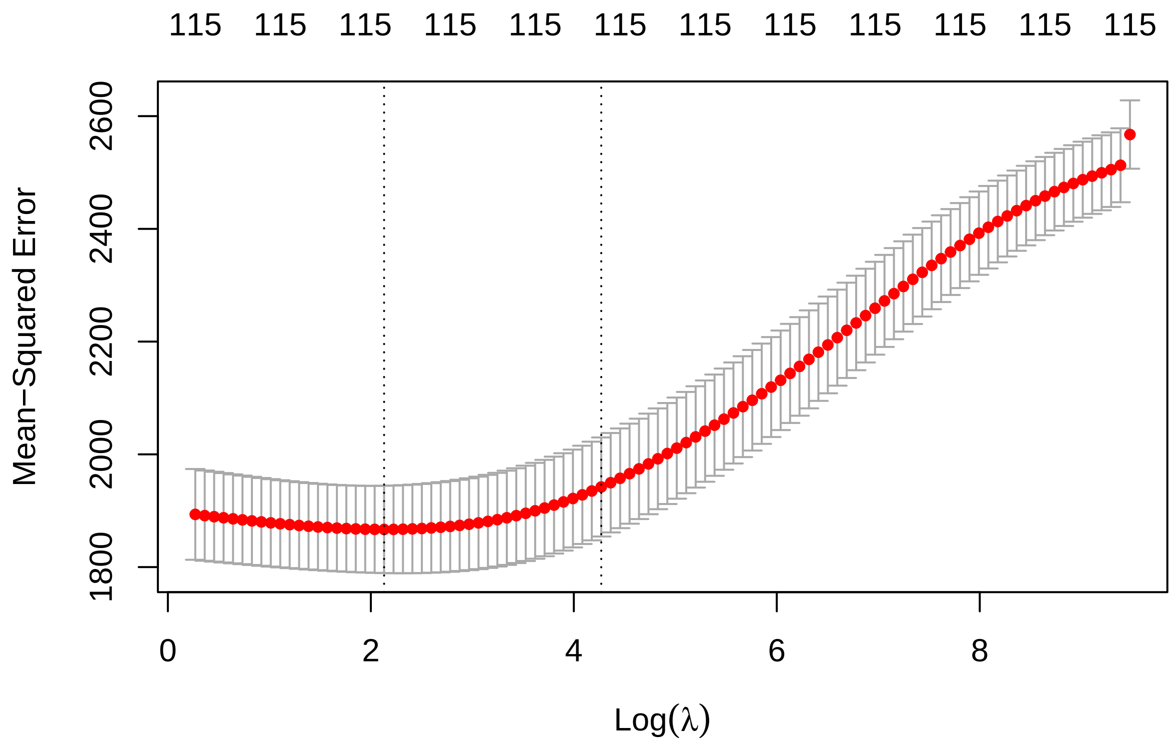
[Here has to be added the formula of the ridge regression]

In fact, this is a possibility to fight over-fitting. If lambda is big the model tends to just take the b0 into account. So the predicted value is the mean of the output variable. If lambda is small, then the model tends to be normal non-regularised model, hence the one from the linear regression.

The command glmnet is used to perform the ridge regression. Cross-validation is performed to find the optimal lambda values. In general, it would be possible with glmnet to just calculate one single model for both output variables. But unfortunately, this option is not available when doing the cross-validation. So, again two independent cross-validations are done to get two values for lambda, one for each output variable.



115 115 115 115 115 115 115 115 115 115 115 115
Mean−Squared Error
280
300
320
340
360
0
2
4
6
8
Log(λ)



It can be seen that the distributions for lambda are quite similar for both models. Although the MSE for both models differ the optimal lambda is quite similar:

```
## [1] 10.83272
```

```
## [1] 8.419139
```

The model performs better with ridge regression as with the baseline. In this model two lambdas are used.

```
## [1] 4492.287
```

A second result is calculated with just one lambda. This lambda is calculated as the mean of the before calculated values of lambda. The result for the adapted version is just slightly worse than with the model with two independent lambdas:

```
## [1] 4496.877
```

Algorithm 2

Short introduction of the second algorithm. What does it do? What are the strengths? What are weaknesses? How is it implemented, including major code snippets.

Results

Here some tables, summaries or especially graphs should be shown here. Maybe this section should be separated into two to show the algorithms for themselves

Discussion

Here follows the discussion of the results. What are the major findings? How did the algorithms perform? Which one was better overall? Is it always better or were the findings which were better by the other one? Which one should be implemented? How could the algorithm be tweeked to perform even better? Where were the problems during implementation? Where are the limits for the algorithms?

How precise do we predict the cities? How far is the difference in kilometres? The authors of the paper where the dataset comes from have a mean great circle error of 3113km? Are we above or below and by how much?

Conclusion

At final some conclusions about the key findings and which algorithm should be used. What was the goal? Were and how were they achieved?

RMarkdown default stuff - needs to be removed but serves up to now as draft

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

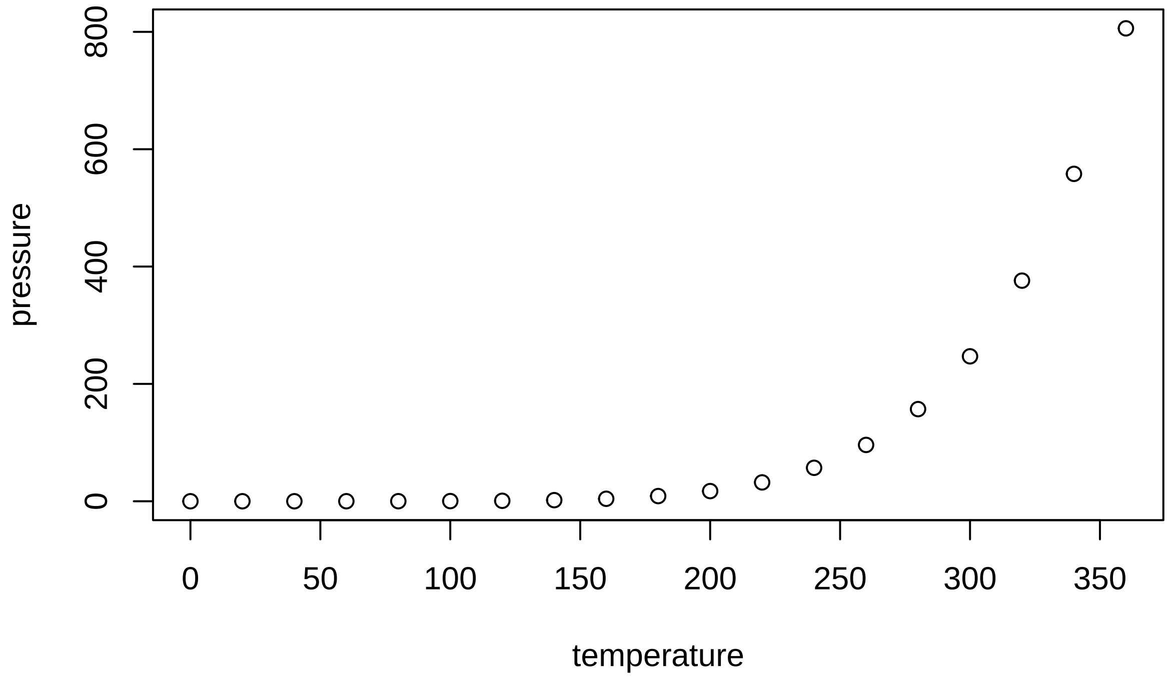
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.