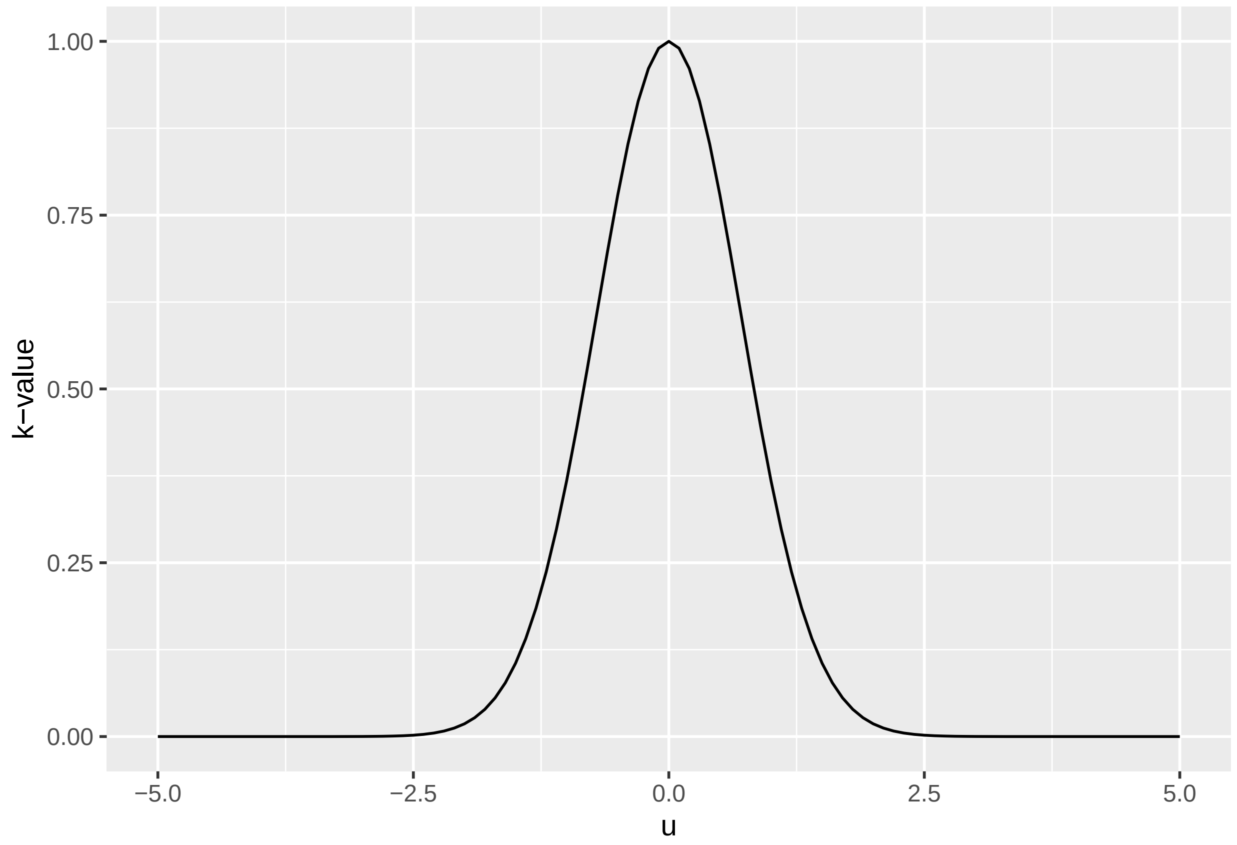
Lab 3 - Group B12

Axel Holmberg (axeho681), Jonathan Reimertz (jonre639) and Wilhelm Hansson (wilha431)

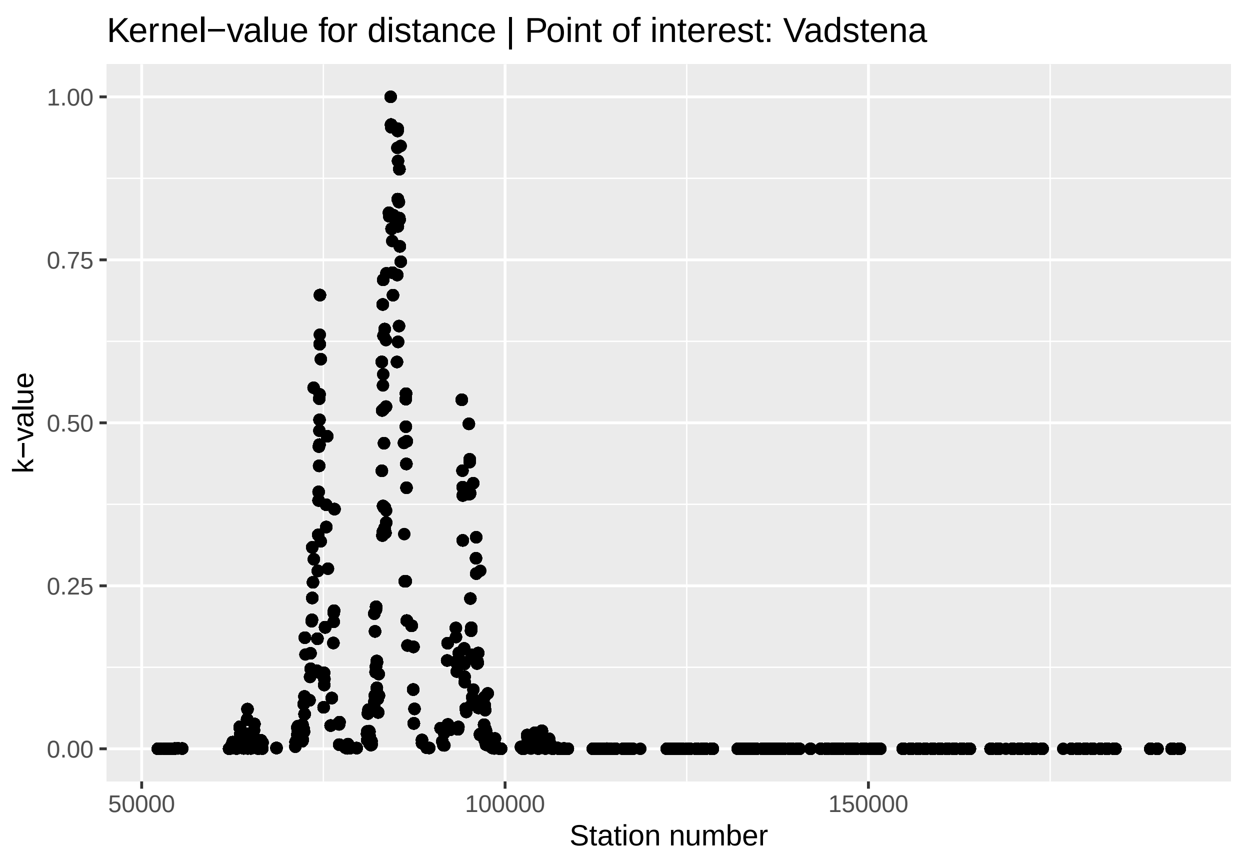
Assignment 1

In this assignment we implemented a kenrel method to predict hourly temperatures for a date and place in Sweden. For this we used a kernel that is the sum of three Gaussian kernels. 1. Distance from observed weather station to point of interest. 2. Days between observed date and date of interest (days of the year). 3. Hours between observed time and time of interest.

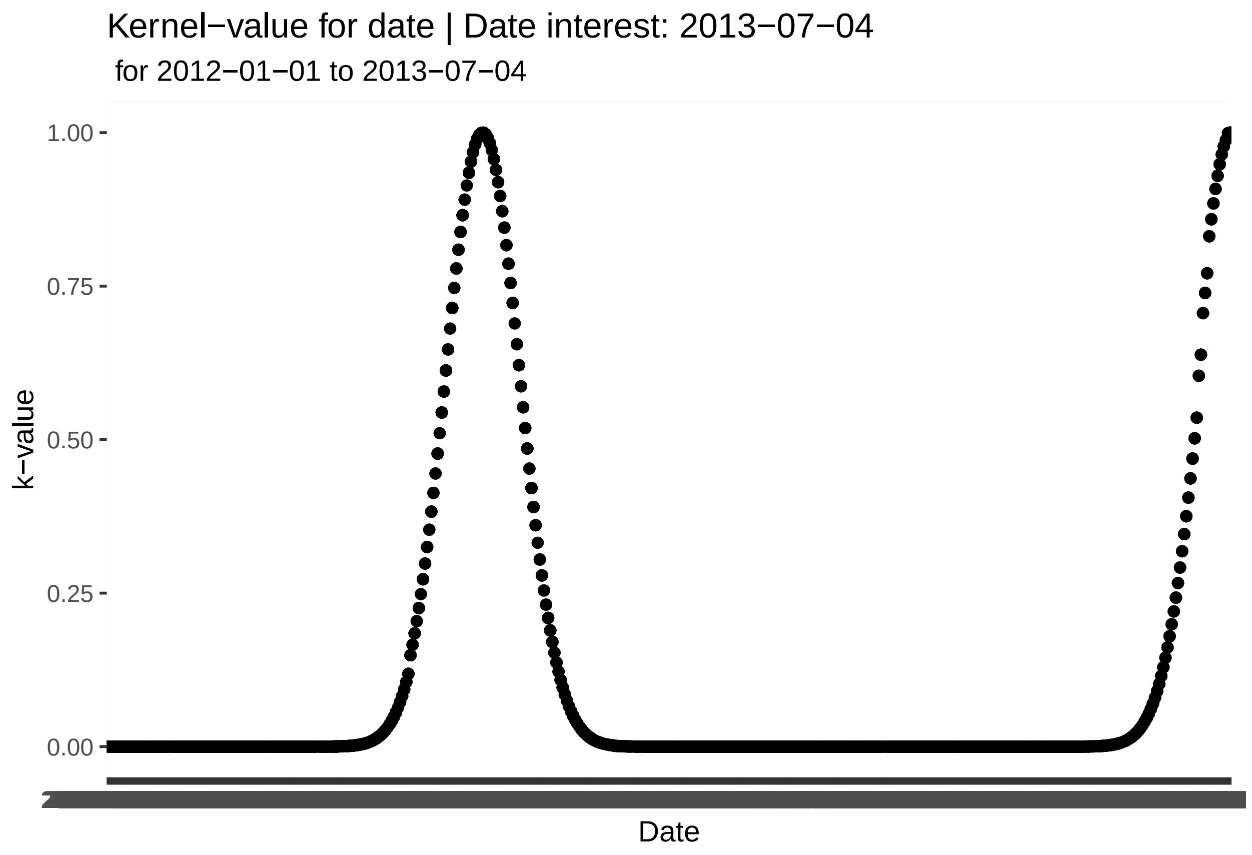
The smoothing coefficients for the three gaussian kernels were chosen so that it they would look similar to the function of $k(u) = e^{-u^2}$, as of graph below.

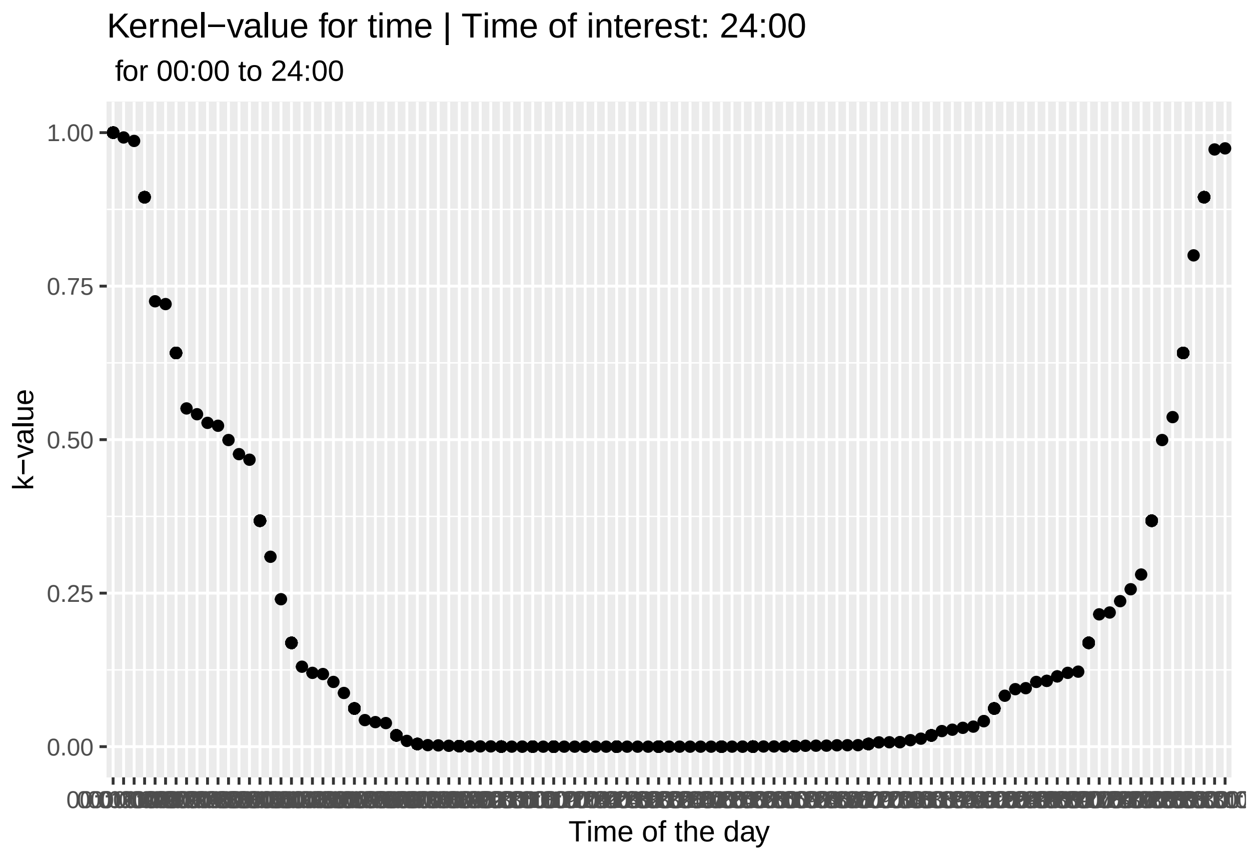


When predicting the temperature for 2013-04-07 in Vadstena (Latitude: 58.4274, Longitude: 14.826) the following plots of the k-value is obtained.



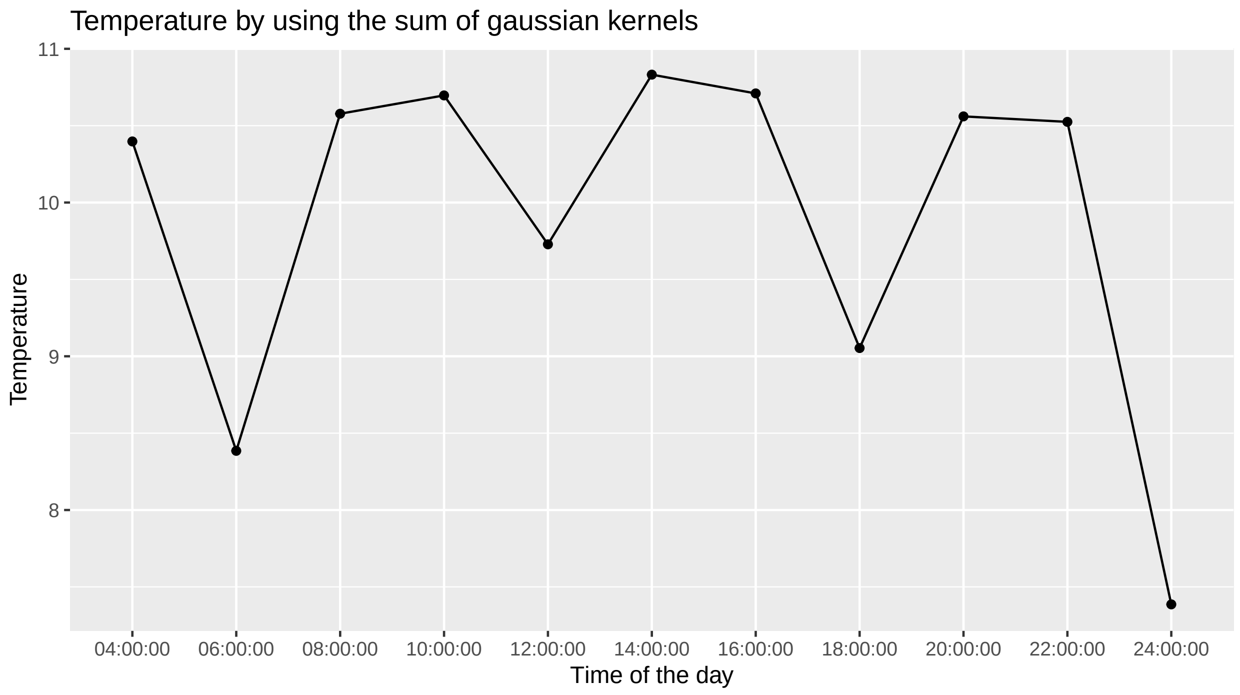
Kernel−value for distance | Point of interest: Vadstena
k−value
1.00
0.75
0.50
0.25
0.00
50000
100000
150000
Station number

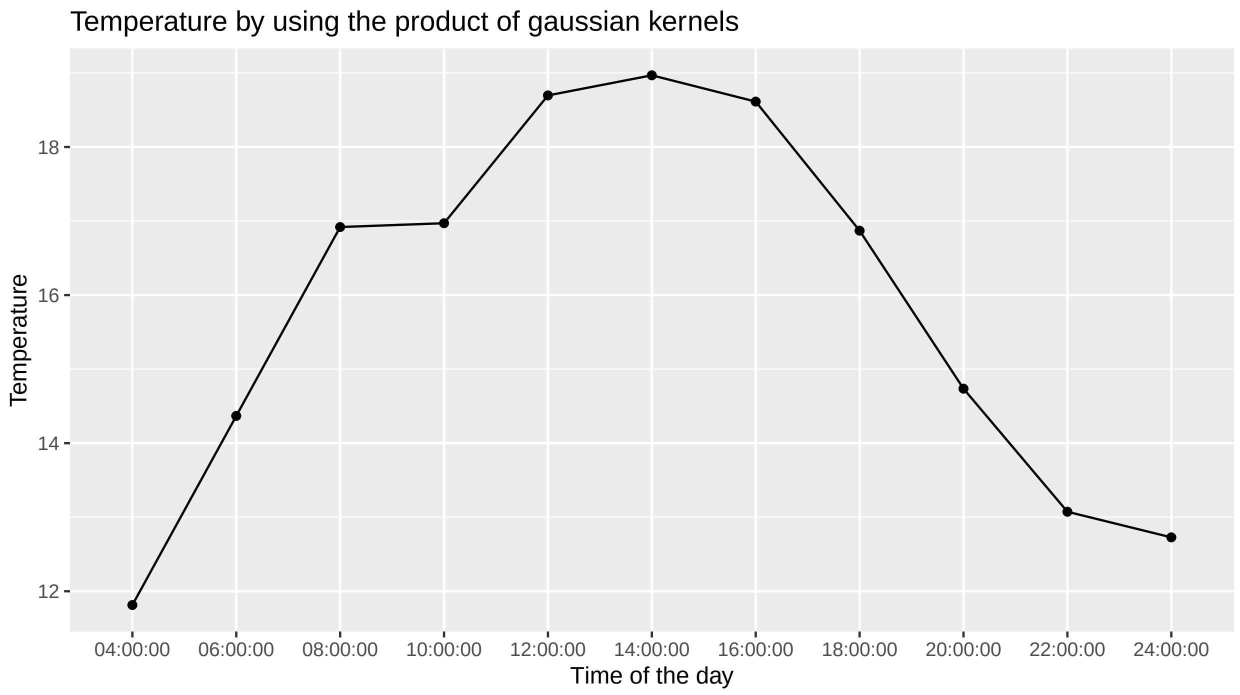

Kernel−value for date | Date interest: 2013−07−04
for 2012−01−01 to 2013−07−04
k−value
1.00
0.75
0.50
0.25
0.00
Date



As visualised by the date-plot, the k-value is dependent on how close the observed date is to the date of interest - this is by the year, which means that for example 5th of August 2019 is equally much accounted for as 5th of Augsut 2018.

When predicting the temperature for 2013-04-07 in Vadstena (same as above), the following graphs were obtained. The first graph by using the sum of the three gaussian kernels and second by using the product of the three gaussian kernels.



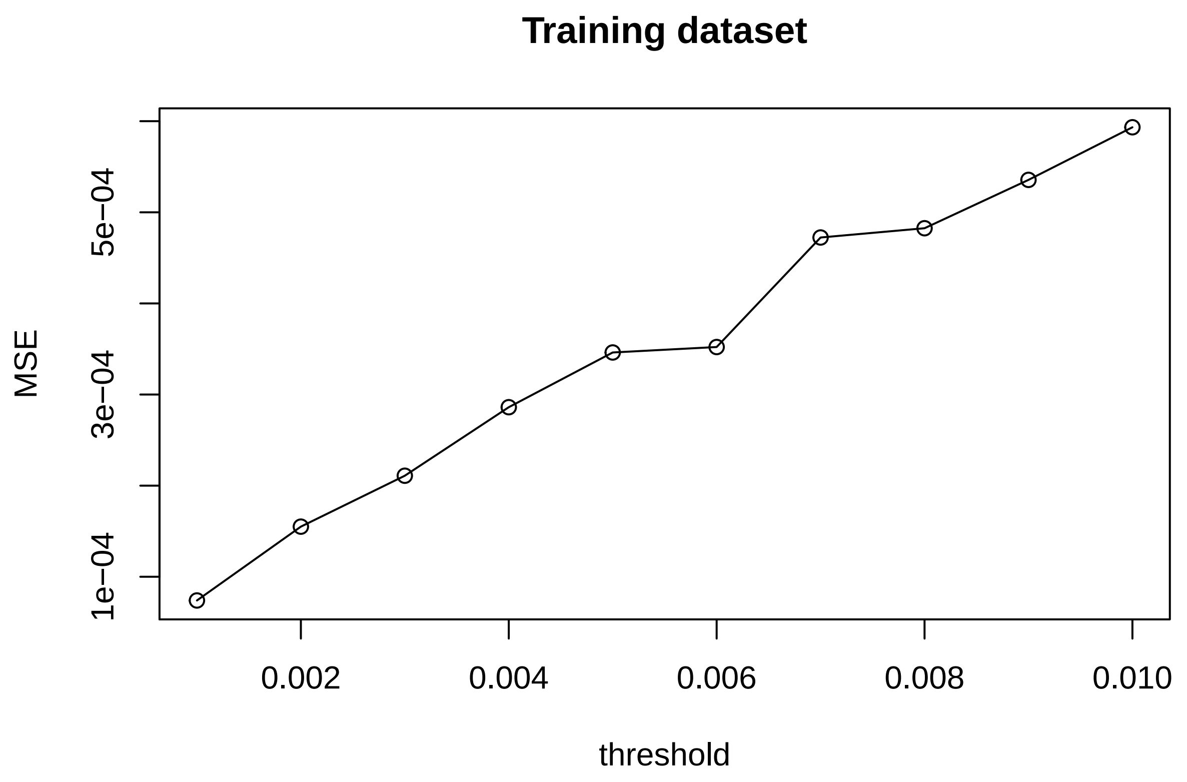


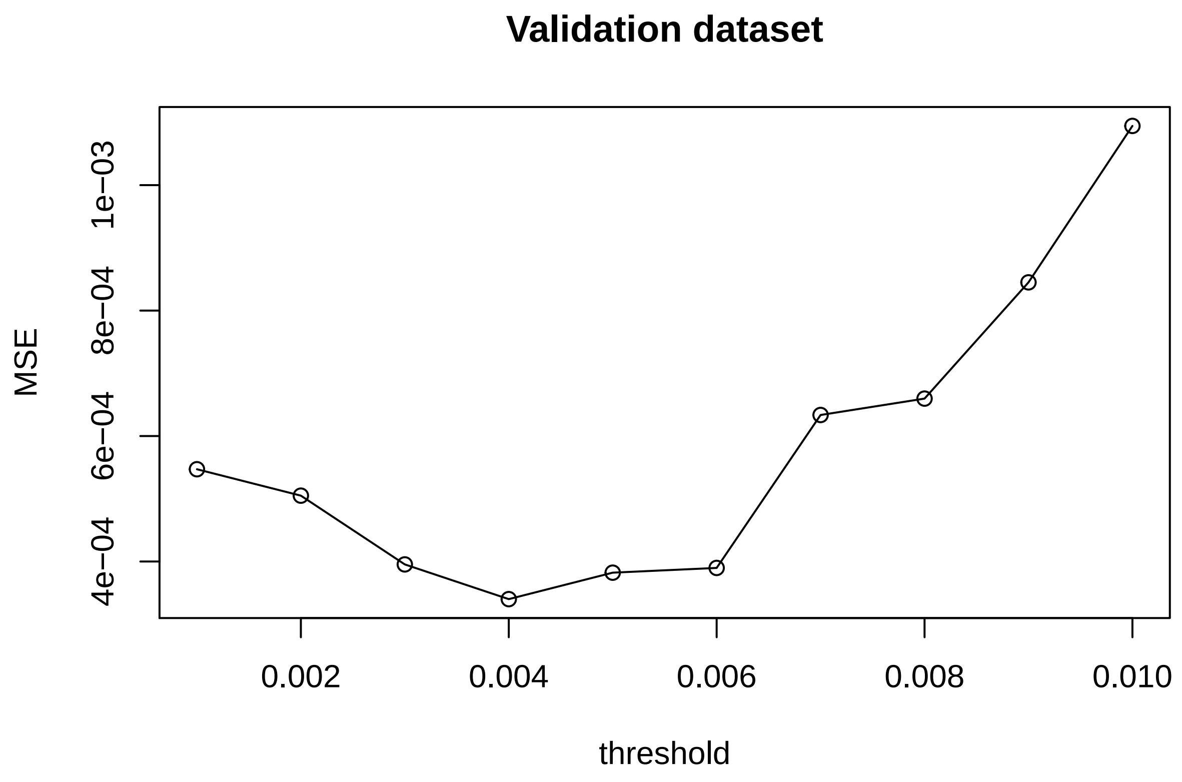
The second plot is more even is by the look of it more correct (comapred to regular weather predictions). This may be because when using the sum of the gaussian kernels the correct values are not as greatly differenting from the "incorrect values". On the contrary, when multiplying the gaussian kernels the correct values gives much greater input than the incorrect values.

This could be handled by choosing more sensitive coefficient for the sum-model.

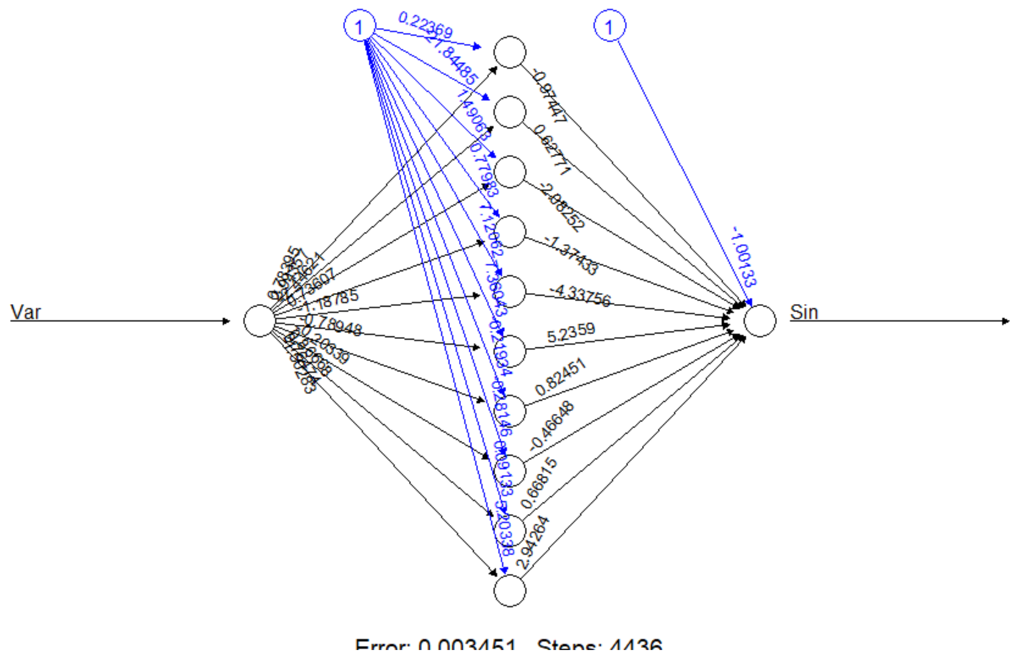
Assignment 3 - Neural network for the sinus function

Assignment 3 is about implementing a neural network for the sinus function based on 50 data points and then choosing the best

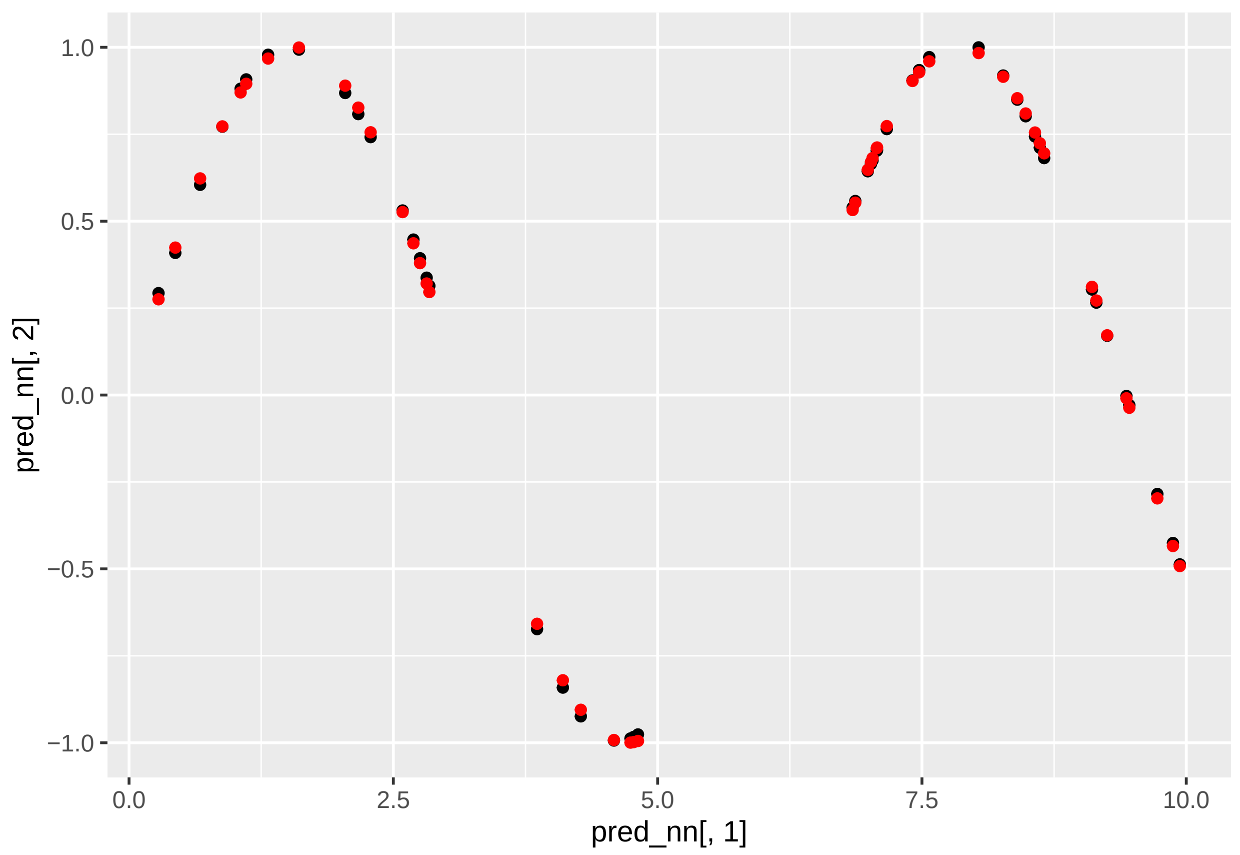




The plots above shows the MSE for the validaiton dataset and the training dataset. One can see that the MSE is lowest for $i = 4$ in the validation plot, and is thereby chosen for the neural network.



Above the neural net can be seen with its' 10 hidden nodes and its' acoompanying values.



Finally the plot with the predicted data (black points) plotted against the original data(red points). As one can see the points almost perfectly line up, which is the desired outcome.

Appendices

Appendix I - Assignment 1

```r
set.seed(1234567890)
library(geosphere) #distHaversine-function
library(ggplot2)

stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations, temps, by="station_number")

################# FUNCTIONS #################

#Filter dates
filterDate <- function(data, filterDate) {
  return (data[!as.Date(data$date) > as.Date(filterDate),])
}

#Filter time
filterTime <- function(data, filterDate, filterTime) {
  return = (data[!(as.Date(k$data.date) == as.Date(filterDate) &
                     strptime(k$data.time, format = "%H:%M:%S") >= strptime(filterTime, format = "%H:%M
}

gaussianKernel <- function(X_Xn, h_value){
  u_value <- X_Xn / h_value
  return(exp(-u_value^2))
}

time.difference <- function(timeVector, timeTemp){
  diff.time = difftime(strptime(timeVector, format = "%H:%M:%S"),
                       strptime(timeTemp, format = "%H:%M:%S"))
  diff.time = as.numeric(diff.time/(60))
  diff.time = ifelse(diff.time < -12, 24 + diff.time, -diff.time)
  return(diff.time)
}

date.difference <- function(dateVector, dateTemp){
  diff.date = as.numeric(as.Date(dateVector) - as.Date(dateTemp), unit="days")
  diff.date = diff.date %% 365.25
  diff.date = ifelse(diff.date > 182, 365-diff.date, diff.date)
  return(diff.date)
}

distance.difference <- function(distanceVector, distanceTemp){

}

################# INPUT #################

# Smoothing coeffiecient for the Gaussian kernels
```

```r
h.distance <- 100000
h.date <- 25
h.time <- 3

# The point to predict
latitude <- 58.4274
longitude <- 14.826
point_to_predict = c(longitude, latitude)

# The date to predict
date <- "2013-07-04"


times <- c("04:00:00", "06:00:00", "08:00:00", "10:00:00",
           "12:00:00", "14:00:00", "16:00:00", "18:00:00",
           "20:00:00", "22:00:00", "24:00:00")

temperature <- vector(length=length(times))


################# CODE #################

# Filter data
data.filtered = filterDate(st, date)

data.date = data.filtered$date
data.time = data.filtered$time
data.station = data.filtered$station_number
data.temperature = data.filtered$air_temperature


# Kernel distance
diff.distance = distHaversine(data.frame(data.filtered$longitude,data.filtered$latitude), point_to_predi
k.distance = gaussianKernel(diff.distance, h.distance)


# Kernel date
k.date = gaussianKernel(date.difference(data.filtered$date, date), h.date)

# K data frame
k.time = data.time
k = data.frame(data.date, data.time, data.station, k.distance, k.date, k.time, data.temperature)

# Result vectors
temperature_sum = vector(length = length(times))
temperature_product = vector(length = length(times))


for (i in 1:length(times)){

  # Kernel time
  k$k.time = gaussianKernel(time.difference(data.filtered$time, times[i]), h.time)
  k$temp_sum = k$k.distance + k$k.date + k$k.time
  k$temp_product = k$k.distance * k$k.date * k$k.time

  k.filtered = filterTime(k, date, times[i])

  temperature_sum[i] = sum(k.filtered$temp_sum %*% k.filtered$data.temperature)/sum(k.filtered$temp_sum
  temperature_product[i] = sum(k.filtered$temp_product %*% k.filtered$data.temperature)/sum(k.filtered$
}

# Weight plots
## Comparison
plot.comparison <- ggplot(data = data.frame(x = 0), mapping = aes(x = x)) + labs (x = "u", y = "k-value
fun1 <- function(x) exp(-x^2)
plot.comparison = plot.comparison + stat_function(fun = fun1) + xlim(-5,5)

## Distance, Date and Time
plot.distance <- ggplot(k, aes(x = data.station, y = k.distance)) + geom_point() + labs(title = "Kernel
plot.date <- ggplot(subset(k, as.Date(k$data.date) > as.Date("2012-01-01")), aes(x = data.date, y = k.d
plot.time <- ggplot(k, aes(x = data.time, y = k.time)) + geom_point() + labs(title = "Kernel-value for

################ RESULT ################
result = data.frame(times, temperature_sum, temperature_product)

plot.sum <- ggplot(result, aes(x = times, y = temperature_sum, group=1)) + geom_point() + geom_line() +
plot.product <- ggplot(result, aes(x = times, y = temperature_product, group=1)) + geom_point() + geom_
```

Appendix II - Assignment 3

```
library(neuralnet)
library(ggplot2)

set.seed(1234567890)
#50 values between 0 and 10
Var <- runif(50, 0, 10)

# Create dataset
trva <- data.frame(Var, Sin=sin(Var))

# Divide dataset into training and validation set
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

# Random initialization of the weights in the interval [-1, 1]
# 31 weights are used
winit <- runif(31, -1, 1)

# Function predicting the mean square error
mse <- function(prediction, observation) {
  return (mean((observation - prediction)^2))
}
```

```r
mse_val <- numeric()
mse_train <- numeric()
threshold <- numeric()
m_sq_err <- function(pred, obs) {
  return (mean((observation - prediction)^2))
}

for(i in 1:10) {
  nn <- neuralnet(Sin ~ Var, data = tr, startweights = winit, hidden = c(10),
                  threshold = i/1000)

  pred_train <- compute(nn, covariate=tr)$net.result
  pred_val <- compute(nn, covariate=va)$net.result
  threshold[i] <- i/1000
  mse_val[i] <- mse(pred_val, va$Sin)
  mse_train[i] <- mse(pred_train, tr$Sin)
  print(i)

}

plot(threshold, mse_val, type="o", ylab="MSE", main = "Validation dataset")
plot(threshold, mse_train, type="o", ylab="MSE", main = "Training dataset")

nn <- neuralnet(Sin ~ Var, data = trva, startweights = winit, hidden = c(10),
                     threshold = 4/1000)
plot(nn)

pred_nn <- prediction(nn)$rep1
plot_net <- ggplot() + geom_point(aes(pred_nn[,1], pred_nn[,2])) +
  geom_point(aes(trva$Var, trva$Sin), colour="red")
plot_net
```