TDDE01 - Laboration 3 Block 1

Olof Simander (olosi122), Gustaf Lindgren (gusli281), Anton Jervebo (antje840)

Dec 2022

Statement of Contribution

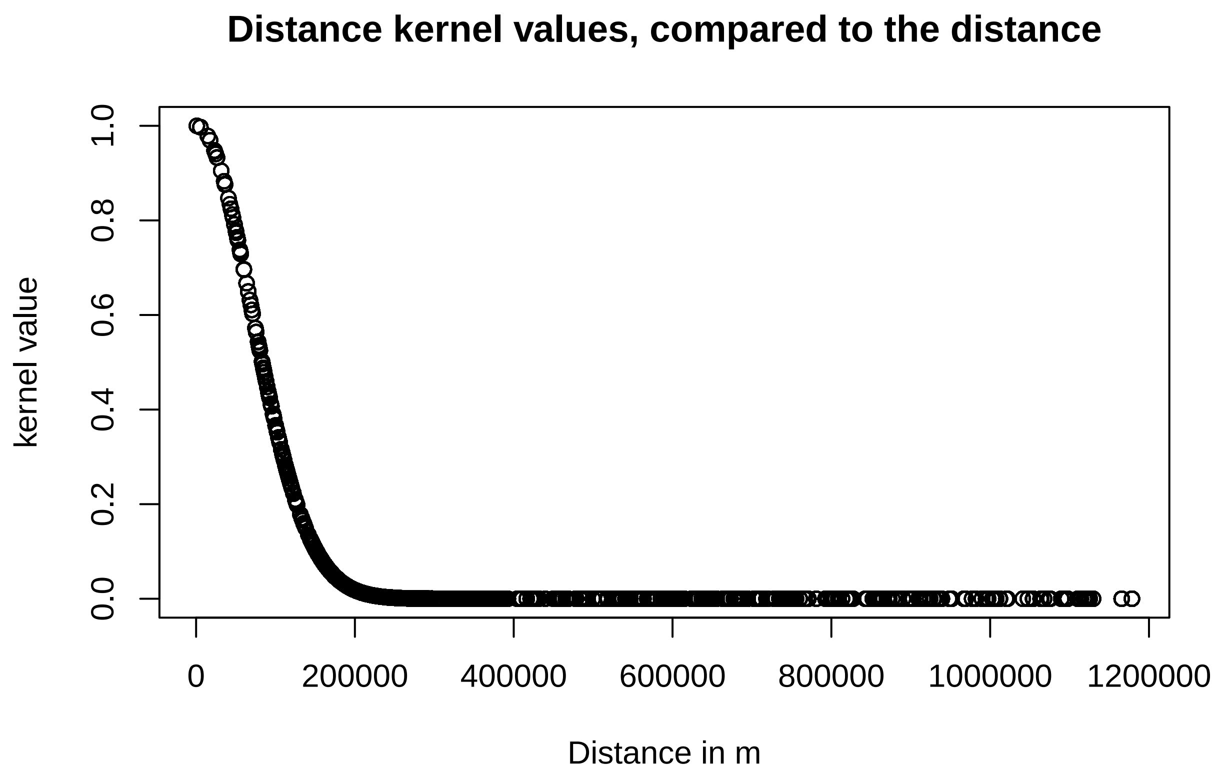
The code and writing of assignment 1 was mainly contributed to by Anton Jervebo. Gustaf Lindgren was responsible for the coding and writing of Assignment 2. Assignment 3 was mainly contributed to by Olof Simander. During various stages of completion, including during the analyses and after completion of writing, the assignments were discussed within the group. These discussions has influenced the report.

Assignment 1

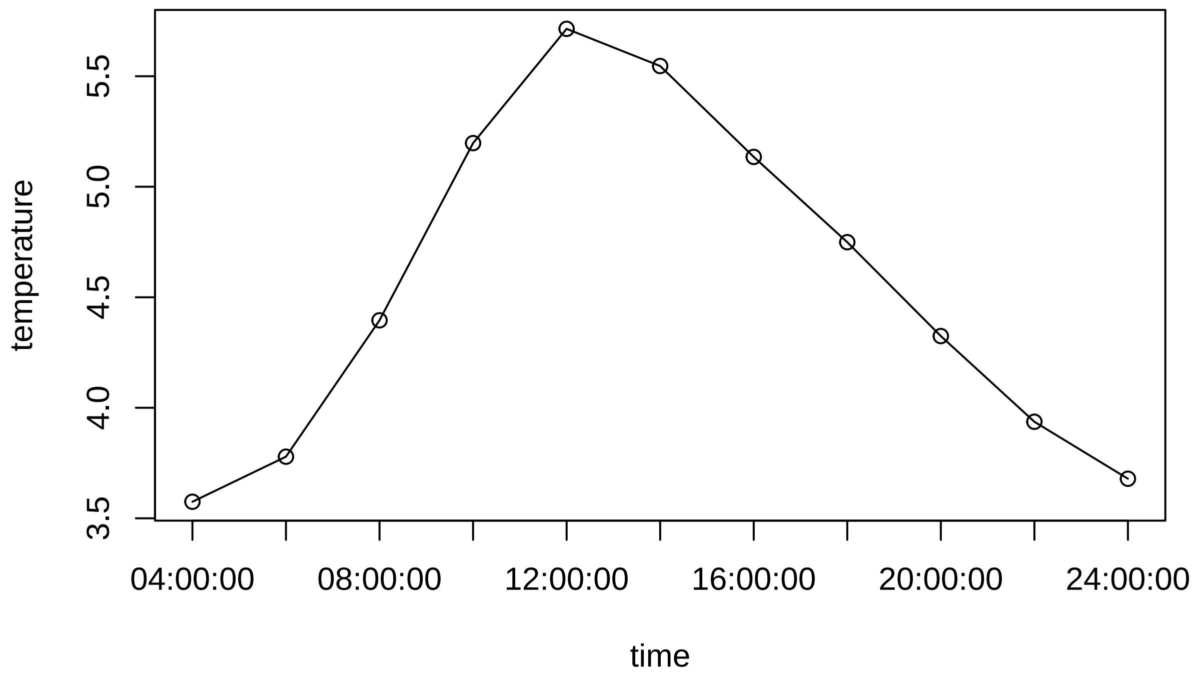
- Create three Gaussian kernels
  - distance, day_difference, hour_difference
- Create a sum of the three kernels
- Create a product of the three kernels
- Elaborate on the reason the may differ

The first step is to choose smoothing factors for the distance, date and time. For the distance between the station and the predicted point, we choose a value to favor the closer measurements over the measurements from stations far from the position we want to predict. In the first plot below, the correlation between distance and kernel value can be seen. For the date, it's feels important to favor the measurement that are closer in time to the date we want to predict, than those that are from a completely different season. Finally, for the smoothing factor for the hours of the day, we chose a smoothing factor that is favoring measurements closer in time of day since the temperatures can fluctuate quite a bit over the day.

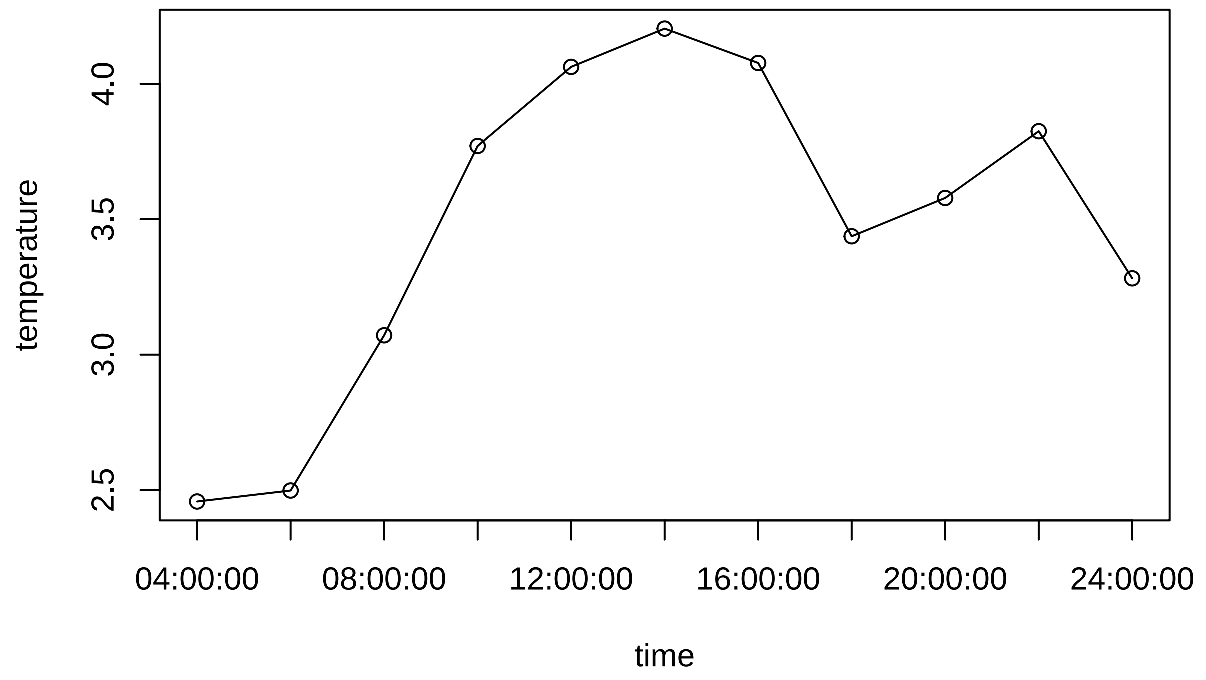
For the predicted position and date, we chose to predict the temperatures in Linköping on 2012-11-03.



The plots below show the predicted temperatures for Linköping at that date. The first plot uses a kernel that is a sum of three Gaussian kernels, and the second one uses a kernel that is the product of three Gaussian kernels.



Looking at the kernel that is a sum of three Gaussian kernels, we can see that the temperatures are starting at a bit cooler than 5 degrees, and are rising to 5.3 degrees at noon. It then predicts the temperatures to steadily decrease for the rest of the day. This doesn't feel like an unreasonable prediction for a November day in Linköping.



Looking at the plot that is the product of three Gaussian kernels, we can see that it's a bit different compared to the previous plot. The first thing to note is the fact that the temperature is decreasing between 4 and 6 in the morning. This feels reasonable considering the sun isn't up at that time. Other things that we can see is that the temperatures are lower than in the previous plot and that the decrease in temperature seems to stop at 17:00, and even starts rising again.

The method to calculate the air temperature is using the score from the three kernels as weights. When using the sum of three Gaussian kernels, the weight has the possibility to be high as long as the point and time we want to predict is close in either distance, time of year or time of day. This is not true while using the multiplication, where the weight is close to zero as long as one of the values from the three kernels are close to zero. This means that the temperature prediction is highly dependent on the physical distance and difference in time of day or year when using the multiplication method.

Appendix A

```r
knitr::opts_chunk$set(echo = TRUE)

set.seed(1234567890)
library(geosphere)

# Had to save the stations.csv with the correct encoding (UTF-8) to be able to read it
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
```

```r
st <- merge(stations, temps, by = "station_number")
h_distance <- 100000 # These three values are up to the students
h_date <- 10
h_time <- 2
a <- 58.41086 # The point to predict (up to the students)
b <- 15.62157

prediction_point <- t(c(b, a))

pred_date <- "2012-11-03" # The date to predict (up to the students)
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"
  )
temp_sum <- vector(length = length(times))
temp_prod <- vector(length = length(times))

# Students' code here

# Filtering measurements on date
st$date <- as.Date(st$date, format = "%m/%d/%Y")
st_filtered <- subset(st, st$date < as.Date(pred_date))

dist_kernel <- function(measured_pos, pred_pos) {
  distance <- distHaversine(measured_pos, pred_pos)
  plot(
    distance,
    exp(-(distance / h_distance) ^ 2),
    type = "p",
    main = "Distance kernel values, compared to the distance",
    xlab = "Distance in m",
    ylab = "kernel value"
  )
  return (exp(-((
    distance / (h_distance)
  ) ^ 2)))
}

day_kernel <- function(measured_date, pred_date) {
  date_diff <-
    as.numeric(as.Date(measured_date) - as.Date(pred_date)) %% 365
  date_diff <-
    ifelse(date_diff < 365 / 2, date_diff, 365 - date_diff)
  return (exp(-((
    date_diff / (h_date)
  ) ^ 2)))
}

hour_kernel <- function(measured_hour, pred_hour) {
  hour_diff <-
    as.numeric(abs(difftime(
      strptime(measured_hour, format = "%H:%M:%S"),
      strptime(pred_hour, format = "%H:%M:%S"),
      units = "hours"
    )))
  hour_diff <- ifelse(hour_diff <= 12, hour_diff, 24 - hour_diff)
  return (exp(-((
    hour_diff / (h_time)
  ) ^ 2)))
}


measured_positions <-
  data.frame(as.numeric(st_filtered$longitude),
             as.numeric(st_filtered$latitude))


dist_kernel_score <-
  dist_kernel(measured_positions, prediction_point)
day_kernel_score <- day_kernel(st_filtered$date, as.Date.character(pred_date))

hour_kernel_score <-
  matrix(nrow = nrow(st_filtered), ncol = length(times))

col <- 1
for (time in times) {
  hour_kernel_score[, col] <- hour_kernel(st_filtered$time, time)
  col <- col + 1
}

# sum of kernels
for (i in 1:length(times)) {
  k_sum <-
    dist_kernel_score + day_kernel_score + hour_kernel_score[, i]
  k_sum <- k_sum / sum(k_sum)
  temp_sum[i] <- sum(k_sum %*% st_filtered$air_temperature)
}

plot(
  temp_sum,
  type = "o",
  xlab = "time",
  ylab = "temperature",
  xaxt = "n"
)
axis(1, at = 1:length(times), labels = times)
```

```r
# mult of kernels
for (i in 1:length(times)) {
  k_prod <-
    dist_kernel_score * day_kernel_score * hour_kernel_score[, i]
  k_prod <- k_prod / sum(k_prod)
  temp_prod[i] <- sum(k_prod %*% st_filtered$air_temperature)
}

plot(
  temp_prod,
  type = "o",
  xlab = "time",
  ylab = "temperature",
  xaxt = "n"
)
axis(1, at = 1:length(times), labels = times
)
```