Q1Block1Lab3

Ravinder Reddy Atla

12/14/2020

```r
set.seed(1234567890)
library(geosphere)
```

```r
# Cleaning Station data
stations <- read.table("stations.csv",sep=',')
Encoding(stations$V2) <- 'latin1'
colnames(stations) <- stations[1,]
stations <- stations[-1,]

temps <- read.csv("temps50k.csv")

# Merged stations and temperature data
st_org <- merge(stations,temps,by="station_number")
```

```r
h_distance <- 102345
h_date <- 2345
h_time <- 10
a <- 58.4274
b <- 14.826
date <- "2013-11-04" # The date to predict (up to the students)
times <- c("04:00:00", "06:00:00",  "8:00:00", "10:00:00","12:00:00",
           "14:00:00", "16:00:00", "18:00:00", "20:00:00","22:00:00",
           "24:00:00")
```

```r
filter_out_posterior <- function(st_org,date,time){
  filtered_data <- subset(st_org,st_org[,'date'] < date | st_org[,'date'] == date &
                            (st_org[,'time']) <= (time))
  return(filtered_data)
}
st <- filter_out_posterior(st_org,date,'08:00:00')
```

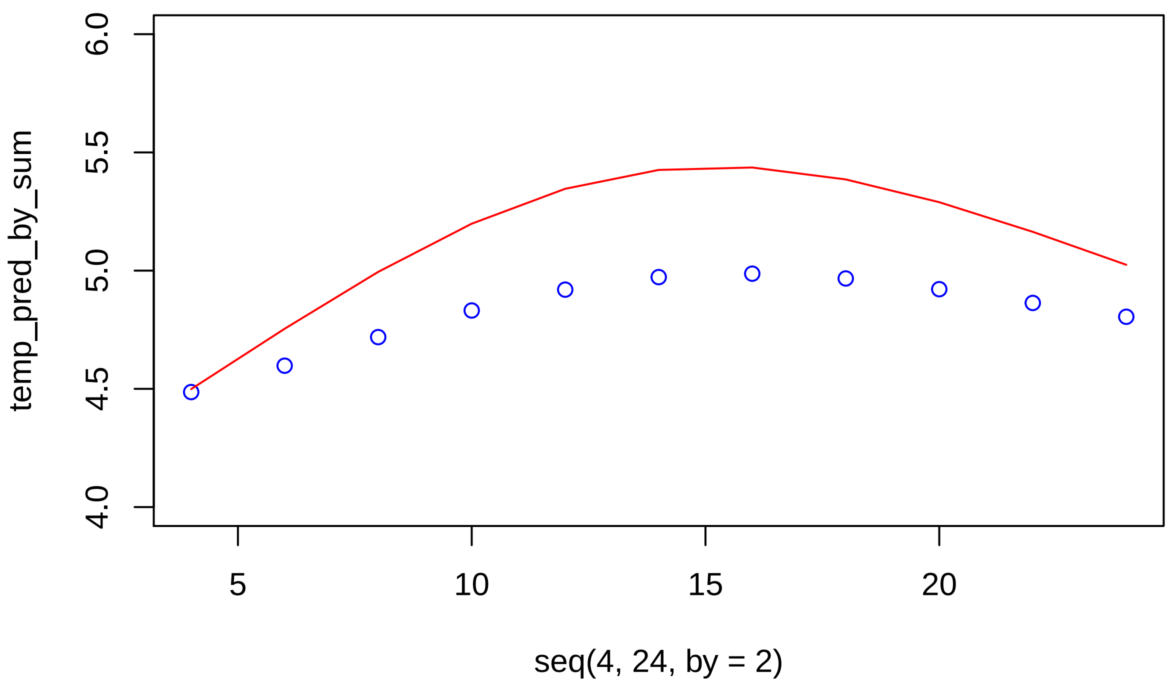
```r
kernel_stations_dist <- function(my_data, a, b, h){
  coord_matrix <- as.matrix(my_data[,c(' longitude',' latitude')])
  mode(coord_matrix) <- 'numeric'
  coord_input_matrix <- matrix(rep(c(a,b),nrow(my_data)),ncol=2, byrow = TRUE)
  distance_station <- distHaversine(coord_matrix, coord_input_matrix, h)
  gaussian_kernel_dist <- exp(-1 * (distance_station/h) ^ 2)
  return(list(d = distance_station, k = gaussian_kernel_dist))
}
```

```r
kernel_day_dist <- function(my_data, date, h){
  date_from_data <- as.Date(my_data[,'date'])
  date_from_input <- as.Date(rep(date, nrow(my_data)))
  distance_days <- abs(as.numeric(difftime(date_from_data,date_from_input, units = 'days')))
  gaussian_kernel_days <- exp(-1 * ((distance_days)/h) ^ 2)
  return(list(d = distance_days, k = gaussian_kernel_days))
}
```

```r
kernel_time_dist <- function(time_data, time_input, h){
  time_data <- as.numeric(unlist(strsplit(time_data,':')))
  time_input <- as.numeric(unlist(strsplit(time_input,':')))

  #Time difference in hours
  time_data_hrs <- time_data[[1]] + (time_data[[2]]/60) + (time_data[[3]]/3600)
  time_input_hrs <- time_input[[1]] + (time_input[[2]]/60) + (time_input[[3]]/3600)
  distance_time <- round(abs(time_data_hrs - time_input_hrs))
  gaussian_kernel_time <- exp(-1 * (distance_time/h) ^ 2)
  return(list(d = distance_time,k = gaussian_kernel_time))
}
```

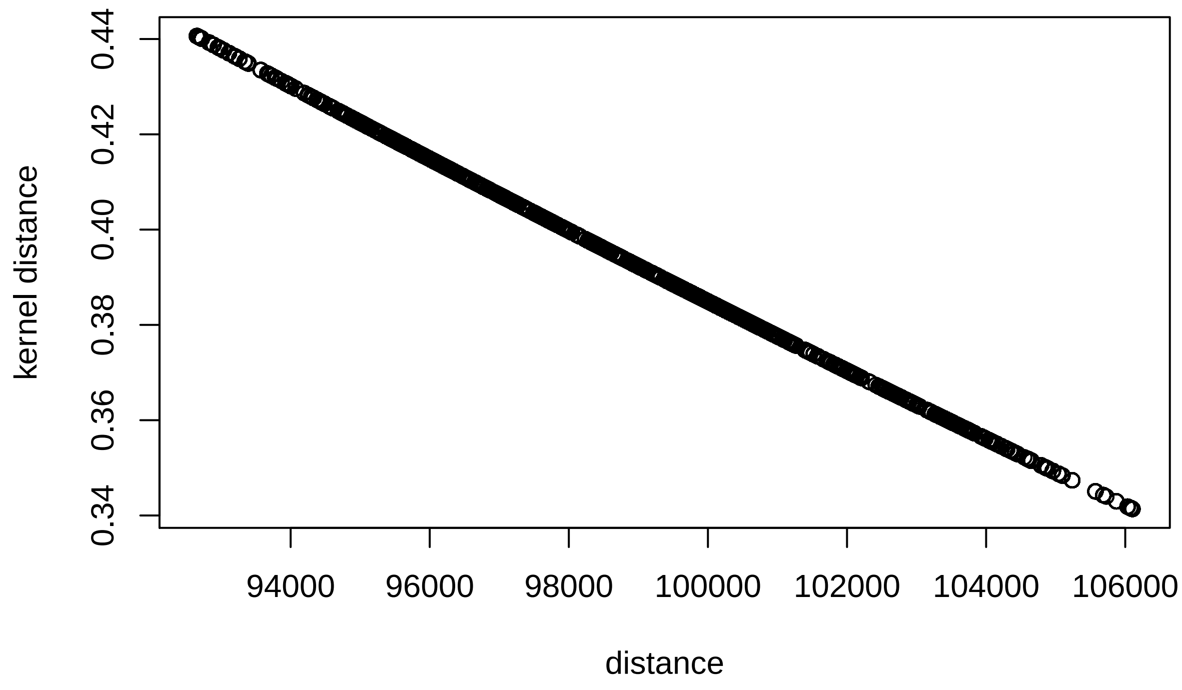
```r
temperature_forecast <- function(my_data,a,b,h_distance,h_date,h_time,
                                    date,times){
    stat_dist <- kernel_stations_dist(my_data,a,b,h_distance)[[2]]
    day_dist <- kernel_day_dist(my_data, date, h_date)[[2]]

    time_dist <- list()
    temperature_pred_sum <- vector()
    temperature_pred_prod <- vector()
    for(a_time in 1:length(times)){
      temp<- lapply(my_data[,'time'],
                                  function(td) kernel_time_dist(td,times[a_time],
                                                           h_time))
      time_dist[[a_time]] <- unlist(temp)[seq(2, nrow(my_data) * 2, by = 2)]

      # Sum of Gaussian Kernel as kernel
      kernel_sum <- stat_dist + day_dist + time_dist[[a_time]]
      temperature_pred_sum[a_time] <- sum(kernel_sum * my_data[,'air_temperature']) /
        sum(kernel_sum)

      # Product of Gaussian Kernels as kernel
      kernel_prod <- stat_dist * day_dist * time_dist[[a_time]]
      temperature_pred_prod[a_time]<- sum(kernel_prod * my_data[,'air_temperature']) /
        sum(kernel_prod)
    }

    return(list(s = temperature_pred_sum, p = temperature_pred_prod))
}
```

```r
temp_pred_by_sum <- temperature_forecast(st,a,b,h_distance,h_date,h_time,
                                         date,times)[[1]]

temp_pred_by_prod <- temperature_forecast(st,a,b,h_distance,h_date,h_time,
                                         date,times)[[2]]

plot(x = seq(4,24,by=2), temp_pred_by_sum, col = 'blue',ylim = c(4,6))
lines(x = seq(4,24,by=2), temp_pred_by_prod, col = 'red')
```

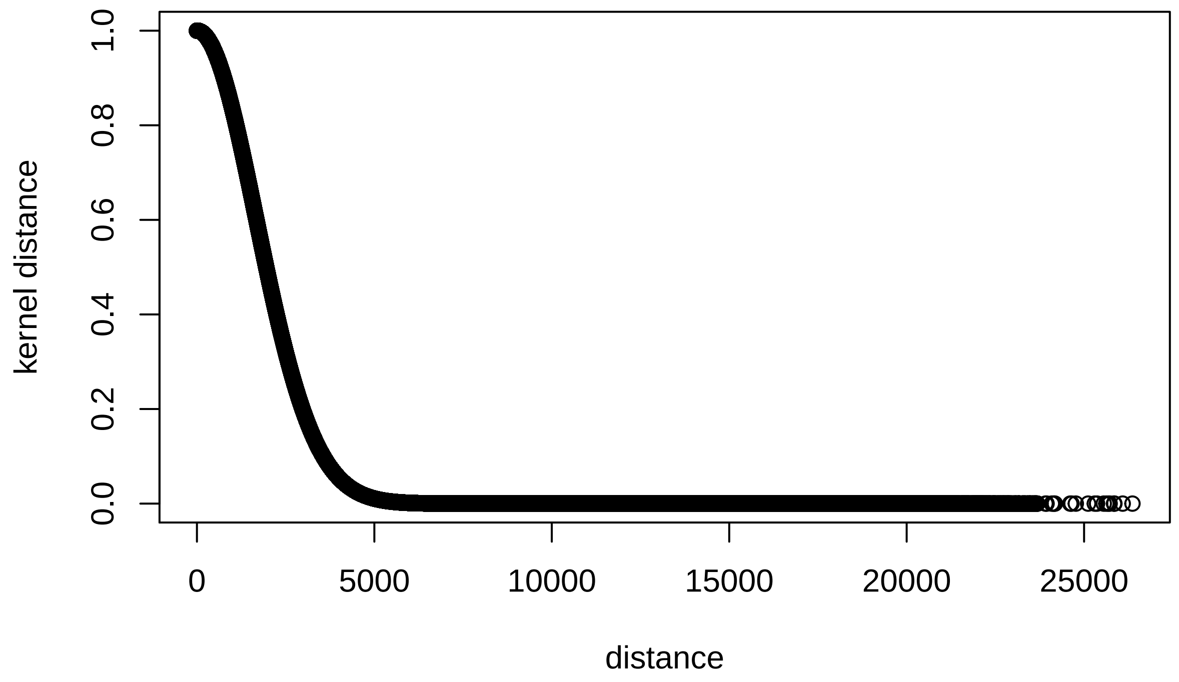


From the above plot, it is evident that temperatures predicted using sum of three Gaussian kernels is less than that of their product. This is because of the difference in the parameters used for predicting the temperature i.e weights and normalizer in the denominator. Weights and normalizer value for product is larger than for sum of kernels.

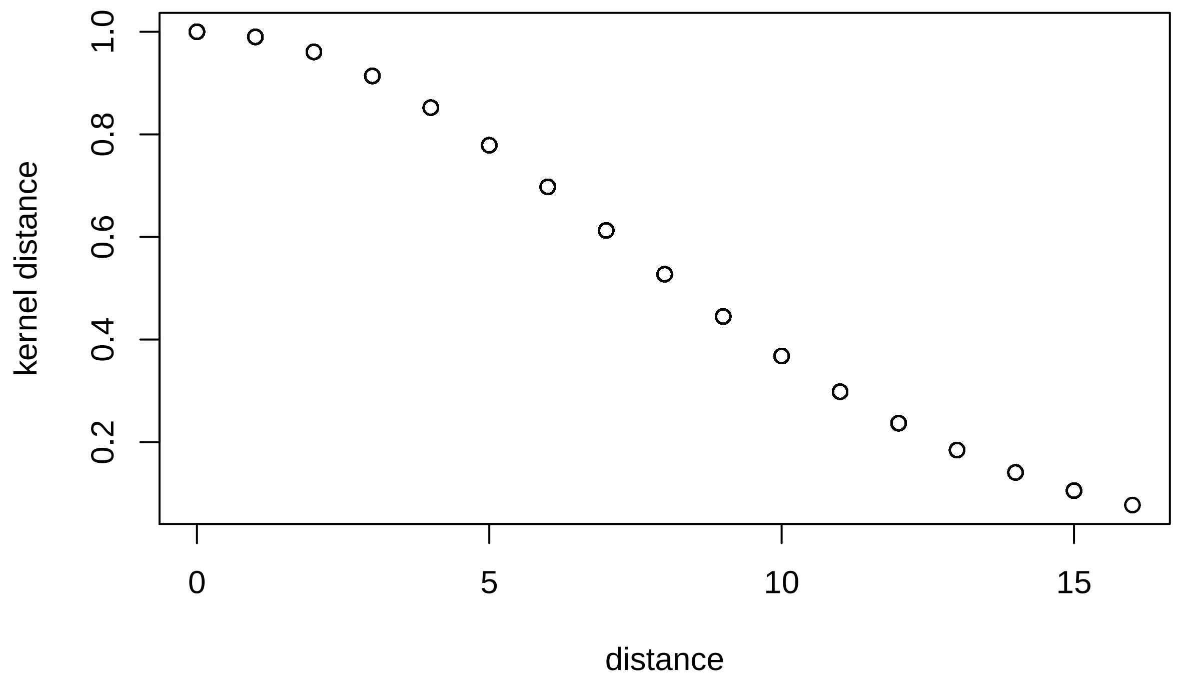
```
plot_station_dist <- plot(kernel_stations_dist(st,a,b,h_distance)[[1]],
                          kernel_stations_dist(st,a,b,h_distance)[[2]],
                          xlab = 'distance', ylab = 'kernel distance')
```



```
plot_day_dist <- plot(kernel_day_dist(st, date, h_date)[[1]],
                      kernel_day_dist(st, date, h_date)[[2]],
                        xlab = 'distance', ylab = 'kernel distance')
```



```
tim <- lapply(st[,'time'],
              function(td) kernel_time_dist(td,'08:00:00',
                                               h_time))
plot_time_dist <- plot(unlist(tim)[seq(1, nrow(st) * 2, by = 2)],
                       unlist(tim)[seq(2, nrow(st) * 2, by = 2)],
                       xlab = 'distance', ylab = 'kernel distance')
```



Appendix

```r
set.seed(1234567890)
library(geosphere)
stations <- read.table("stations.csv",sep=',')

# Cleaning Station data
Encoding(stations$V2) <- 'latin1'
colnames(stations) <- stations[1,]
stations <- stations[-1,]

temps <- read.csv("temps50k.csv")

# Merged stations and temperature data
st_org <- merge(stations,temps,by="station_number")


filter_out_posterior <- function(st_org,date,time){
  filtered_data <- subset(st_org,st_org[,'date'] < date | st_org[,'date'] == date &
                            (st_org[,'time']) <= (time))
  return(filtered_data)
}

h_distance <- 102345# These three values are up to the students
h_date <- 2345
h_time <- 10
a <- 58.4274 # The point to predict (up to the students)
b <- 14.826
date <- "2013-11-04" # The date to predict (up to the students)
times <- c("04:00:00", "06:00:00",  "8:00:00", "10:00:00","12:00:00",
           "14:00:00", "16:00:00", "18:00:00", "20:00:00","22:00:00",
           "24:00:00")

st <- filter_out_posterior(st_org,date,'08:00:00')
#temp <- vector(length=length(times))

# Students' code here
kernel_stations_dist <- function(my_data, a, b, h){
  coord_matrix <- as.matrix(my_data[,c(' longitude',' latitude')])
  mode(coord_matrix) <- 'numeric'
  coord_input_matrix <- matrix(rep(c(a,b),nrow(my_data)),ncol=2, byrow = TRUE)
  distance_station <- distHaversine(coord_matrix, coord_input_matrix, h)
  gaussian_kernel_dist <- exp(-1 * (distance_station/h) ^ 2)
  return(list(d = distance_station, k = gaussian_kernel_dist))
}

kernel_day_dist <- function(my_data, date, h){
  date_from_data <- as.Date(my_data[,'date'])
  date_from_input <- as.Date(rep(date, nrow(my_data)))
  distance_days <- abs(as.numeric(difftime(date_from_data,date_from_input, units = 'days')))
  gaussian_kernel_days <- exp(-1 * (as.numeric(distance_days)/h) ^ 2)
  return(list(d = distance_days, k = gaussian_kernel_days))
}
```

```r
kernel_time_dist <- function(time_data, time_input, h){
  time_data <- as.numeric(unlist(strsplit(time_data,':')))
  time_input <- as.numeric(unlist(strsplit(time_input,':')))

  #Time difference in hours
  time_data_hrs <- time_data[[1]] + (time_data[[2]]/60) + (time_data[[3]]/3600)
  time_input_hrs <- time_input[[1]] + (time_input[[2]]/60) + (time_input[[3]]/3600)
  distance_time <- round(abs(time_data_hrs - time_input_hrs))
  gaussian_kernel_time <- exp(-1 * (distance_time/h) ^ 2)
  return(list(d = distance_time,k = gaussian_kernel_time))
}


temperature_forecast <- function(my_data,a,b,h_distance,h_date,h_time,
                                    date,times){
    stat_dist <- kernel_stations_dist(my_data,a,b,h_distance)[[2]]

    day_dist <- kernel_day_dist(my_data, date, h_date)[[2]]
    time_dist <- list()
    temperature_pred_sum <- vector()
    temperature_pred_prod <- vector()
    for(a_time in 1:length(times)){
      temp<- lapply(my_data[,'time'],
                              function(td) kernel_time_dist(td,times[a_time],
                                                         h_time))
      time_dist[[a_time]] <- unlist(temp)[seq(2, nrow(my_data) * 2, by = 2)]

      # Sum of Gaussian Kernel as kernel
      kernel_sum <- stat_dist + day_dist + time_dist[[a_time]]
      temperature_pred_sum[a_time] <- sum(kernel_sum * my_data[,'air_temperature']) /
        sum(kernel_sum)

      # Product of Gaussian Kernels as kernel
      kernel_prod <- stat_dist * day_dist * time_dist[[a_time]]
      temperature_pred_prod[a_time]<- sum(kernel_prod * my_data[,'air_temperature']) /
        sum(kernel_prod)
    }

    return(list(s = temperature_pred_sum, p = temperature_pred_prod))
}

temp_pred_by_sum <- temperature_forecast(st,a,b,h_distance,h_date,h_time,
                                         date,times)[[1]]

temp_pred_by_prod <- temperature_forecast(st,a,b,h_distance,h_date,h_time,
                                         date,times)[[2]]

plot(x = seq(4,24,by=2), temp_pred_by_sum, col = 'blue',ylim = c(4,6))
lines(x = seq(4,24,by=2), temp_pred_by_prod, col = 'red')



plot_station_dist <- plot(kernel_stations_dist(st,a,b,h_distance)[[1]],
                          kernel_stations_dist(st,a,b,h_distance)[[2]],
                          xlab = 'distance', ylab = 'kernel distance')


plot_day_dist <- plot(kernel_day_dist(st, date, h_date)[[2]],
                      kernel_day_dist(st, date, h_date)[[2]],
                      xlab = 'distance', ylab = 'kernel distance')

tim <- lapply(st[,'time'],
              function(td) kernel_time_dist(td,'08:00:00',
                                            h_time))
plot_time_dist <- plot(unlist(tim)[seq(1, nrow(st) * 2, by = 2)],
                       unlist(tim)[seq(2, nrow(st) * 2, by = 2)],
                       xlab = 'distance', ylab = 'kernel distance')
```