```r
#### Lab 3 - Assignment 1 - Kernel Methods
### Libraries
library(geosphere)

### Setup
# data
data.temps = read.csv("temps50k.csv", fileEncoding="latin1")
data.stations = read.csv("stations.csv", fileEncoding="latin1")
data = merge(data.temps, data.stations, by="station_number")

### Functions
## Filter functions
# Returns the data excluding those with dates after given date
get_filtered_data_date = function(data, ..date){
  date = as.Date(..date)
  dates = as.Date(data$date)
  dates.valid = dates <= date
  return(data[dates.valid,])
}

# Returns the data excluding those with time after the time of a given
date
get_filtered_data_time = function(data, date, ..time){
  time = strptime(..time, format="%H")
  times = strptime(data$time, format="%H")
  times.diff = as.numeric(difftime(time, times, units="hours"))
  times.valid = as.Date(data$date)<as.Date(date) | (times.diff < 0)
  return(data[times.valid,])
}

## Distance functions
# Returns distances between one point and other points
get_distance_differences = function(point, points){
  return(distHaversine(point, points))
}

# Returns the time difference between one time and other times
get_time_differences = function(..time, ..times){
  time = strptime(..time, format="%H:")
  times = strptime(..times, format="%H:")
  times.diff = as.numeric(difftime(time,times, units="hours"))
  times.diff = times.diff %% 24
  return(times.diff)
}

# Returns the date difference between one date and other dates
get_date_differences = function(..date, ..dates){
  date = as.Date(..date)
  dates = as.Date(..dates)
  date.diff = as.numeric(difftime(date,dates))
  date.diff = date.diff # %% 365
  return(date.diff)
}
```

```r
## Kernel Functions
# Returns the gaussian kernel value for given euclidian norm of u
get_gaussian_kernel = function(u){
  return(exp(-u^2))
}

# Returns the kernel values for all the distances of given data to a
point
get_gaussian_kernels_distance = function(point, data, h.distance){
  n = dim(data[1])
  points = matrix(c(data$latitude, data$longitude), nrow=n, ncol=2)
  distance.differences = get_distance_differences(point, points)
  u.distances = distance.differences/h.distance
  gaussian.kernels.distance = get_gaussian_kernel(u.distances)
  return(gaussian.kernels.distance)
}

# Returns the kernel values for all the dates of given data to a date
get_gaussian_kernels_date = function(date, data, h.days){
  dates = data$date
  date.differences = get_date_differences(date, dates)
  u.dates = date.differences/h.days
  gaussian.kernels.date = get_gaussian_kernel(u.dates)
  return(gaussian.kernels.date)
}

# Returns the kernel values for all the times of given data to a time
get_gaussian_kernels_time = function(time, data, h.time){
  times = data$time
  time.differences = get_time_differences(time, times)
  u.times = time.differences/h.time
  gaussian.kernels.time = get_gaussian_kernel(u.times)
  return(gaussian.kernels.time)
}

# Returns a prediction using the sum of kernels
y_sum = function(kernels.distance, kernels.date, kernels.time, data){
  temperatures = data$air_temperature
  kernels.sum = kernels.distance + kernels.date + kernels.time
  y.sum = (kernels.sum %*% temperatures) / sum(kernels.sum)
  return(y.sum)
}

# Returns a prediction using the product of kernels
y_prod = function(kernels.distance, kernels.date, kernels.time, data){
  temperatures = data$air_temperature
  kernels.prod = kernels.distance * kernels.date * kernels.time
  y.prod = (kernels.prod %*% temperatures) / sum(kernels.prod)
  return(y.prod)
}

### Implementation
## Task 1 - Find reasonable kernal value
# values to test
xgrid.distance = seq(0,1000000)
xgrid.date = seq(0,31)
xgrid.time = seq(0:24)

# kernel width candidates
h_distance = 50000
h_date = 7
h_time = 3

# Plot the gaussian value for given sequence
plot(xgrid.distance/1000,
get_gaussian_kernel(xgrid.distance/h_distance), type="l",
     main="Gassian Kernal value for different location distances",
xlab="Distance (km)", ylab="Kernal Value")
plot(xgrid.date, get_gaussian_kernel(xgrid.date/h_date), type="l",
     main="Gassian Kernal value for different date distances",
xlab="Date Difference (days)", ylab="Kernal Value")
plot(xgrid.time, get_gaussian_kernel(xgrid.time/h_time), type="l",
     main="Gassian Kernal value for different time distances",
xlab="Time Difference (hours)", ylab="Kernal Value")

## Task 2 - Make predictions using kernel methods

# Place, date and times to make the predictions
lat_stockholm = 58.3979255
long_stockholm = 15.576518
point = c(lat_stockholm, long_stockholm)
date = "2013-01-01"
times = sprintf("%02d:00:00", seq(4, 24, 2))

# Preparation
data.filtered.date = get_filtered_data_date(data, date) # filter data
based on date
y.sum = numeric(length(times)) # to be filled with predictions based
on sum of kernels
y.prod = numeric(length(times)) # to be filled with predictions based
on product of kernels

# Calculation
m = length(times) # number of predictions to be made
for(i in 1:m){
  time = times[i] # time to predict temperature at
  data.filtered.time = get_filtered_data_time(data.filtered.date,
date, time)
  gaussian.kernels.distance = get_gaussian_kernels_distance(point,
data.filtered.time, h_distance)
  gaussian.kernels.date = get_gaussian_kernels_date(date,
data.filtered.time, h_date)
  gaussian.kernels.time = get_gaussian_kernels_time(time,
data.filtered.time, h_time)
  y.sum[i] = y_sum(gaussian.kernels.distance, gaussian.kernels.date,
gaussian.kernels.time, data.filtered.time)
  y.prod[i] = y_prod(gaussian.kernels.distance, gaussian.kernels.date,
gaussian.kernels.time, data.filtered.time)
}
```

```
## Task 3 - Plot the predicted temperatures
# Sum of kernels
plot(y.sum, xaxt='n', main="Predicted Temperature (Sum of kernels)",
ylab="Temperature", xlab="Time")
axis(1, at=1:length(times), labels=times)

# Product of kernels
plot(y.prod, xaxt='n', main="Predicted Temperature (Product of
kernels)", ylab="Temperature", xlab="Time")
axis(1, at=1:length(times), labels=times)
```