In [1]:
```
library('geosphere')
```

In [3]:
```
par(family  = "Arial")
#install.packages("showtext")
library(showtext)
showtext_auto()
options(repr.plot.width=5, repr.plot.height=5)
```

Assignment 1: Kernel Methods

Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files `stations.csv` and `temps50k.csv`. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).

You are asked to provide a temperature forecast for a date and place in Sweden. The forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. Use a kernel that is the **sum** of three Gaussian kernels:

- The first to account for the **physical** distance from a station to the point of interest. For this purpose, use the function `distHaversine` from the R package `geosphere`.
- The second to account for the distance between the **day** a temperature measurement was made and the day of interest.
- The third to account for the distance between the **hour** of the day a temperature measurement was made and the hour of interest.

Choose an appropriate smoothing coefficient or width for each of the three kernels above. No cross-validation should be used. Instead, choose manually a width that gives large kernel values to closer points and small values to distant points. Show this with a **plot** of the kernel value as a function of distance.

Finally, repeat the exercise above by combining the three kernels into one by **multiplying** them, instead of summing them up. Compare the results obtained in both cases and elaborate on why they may differ.

Note that the file `temps50k.csv` may contain temperature measurements that are posterior to the day and hour of your forecast. You must **filter** such measurements out, i.e. they cannot be used to compute the forecast. Feel free to use the template below to solve the assignment.

In [20]:
```
#1 Implement Kernel Method to predict hourly temp/day (4am -24pm)
#station_data <- read.csv("stations.csv",stringsAsFactors=FALSE, fileEncoding="latin1")

library(dplyr)

# Read data
station_data <- read.csv("stations.csv", fileEncoding="latin1")
head(station_data,2)
temp_data <- read.csv("temps50k.csv",fileEncoding="latin1")
head(temp_data,2)
```

station_number	station_name	measurement_height	latitude	longitude	readings_from	readings_to	elevation
102170	Östmark-Åsarna	2	60.2788	12.8538	2013-11-01 00:00:00	2016-09-30 23:59:59	135
102190	Östmark-Lämbacken	2	60.3097	12.6959	1955-09-01 00:00:00	1980-02-29 23:59:59	177

station_number	date	time	air_temperature	quality
151550	1981-02-23	12:00:00	-13.4	G
54230	1990-07-02	00:00:00	15.8	G

In [19]:
```
st <- merge(station_data, temp_data, by = "station_number")
n = dim(st)[1]
head(st,2)
```

station_number	station_name	measurement_height	latitude	longitude	readings_from	readings_to	elevation	date	time	air_temperature	quality
52230	Falsterbo	2	55.3836	12.8203	1949-01-01 00:00:00	2016-10-01 06:00:00	5	2004-05-28	12:00:00	12	G
52230	Falsterbo	2	55.3836	12.8203	1949-01-01 00:00:00	2016-10-01 06:00:00	5	1951-05-10	18:00:00	10	Y

In [60]:
```r
a <- 58.4274 # The point to predict (up to the students)
b <- 14.826
p2 <- c(a,b)
date_pred <- "2003-07-04" # 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 <- vector(length=length(times))
st <- st %>% filter(as.Date(date) < as.Date(date_pred)) #remove the posterior date
```

In [77]:
```r
library(lubridate)
date_df <- as.Date(st$date)
year(date_df) <- 2020
date_pred <- as.Date(date_pred)
year(date_pred) <- 2020
diff_days <- as.numeric(abs(date_pred-date_df))
diff_days <- ifelse(diff_days<183, diff_days, 365-diff_days)
st$diff_days <- diff_days
```

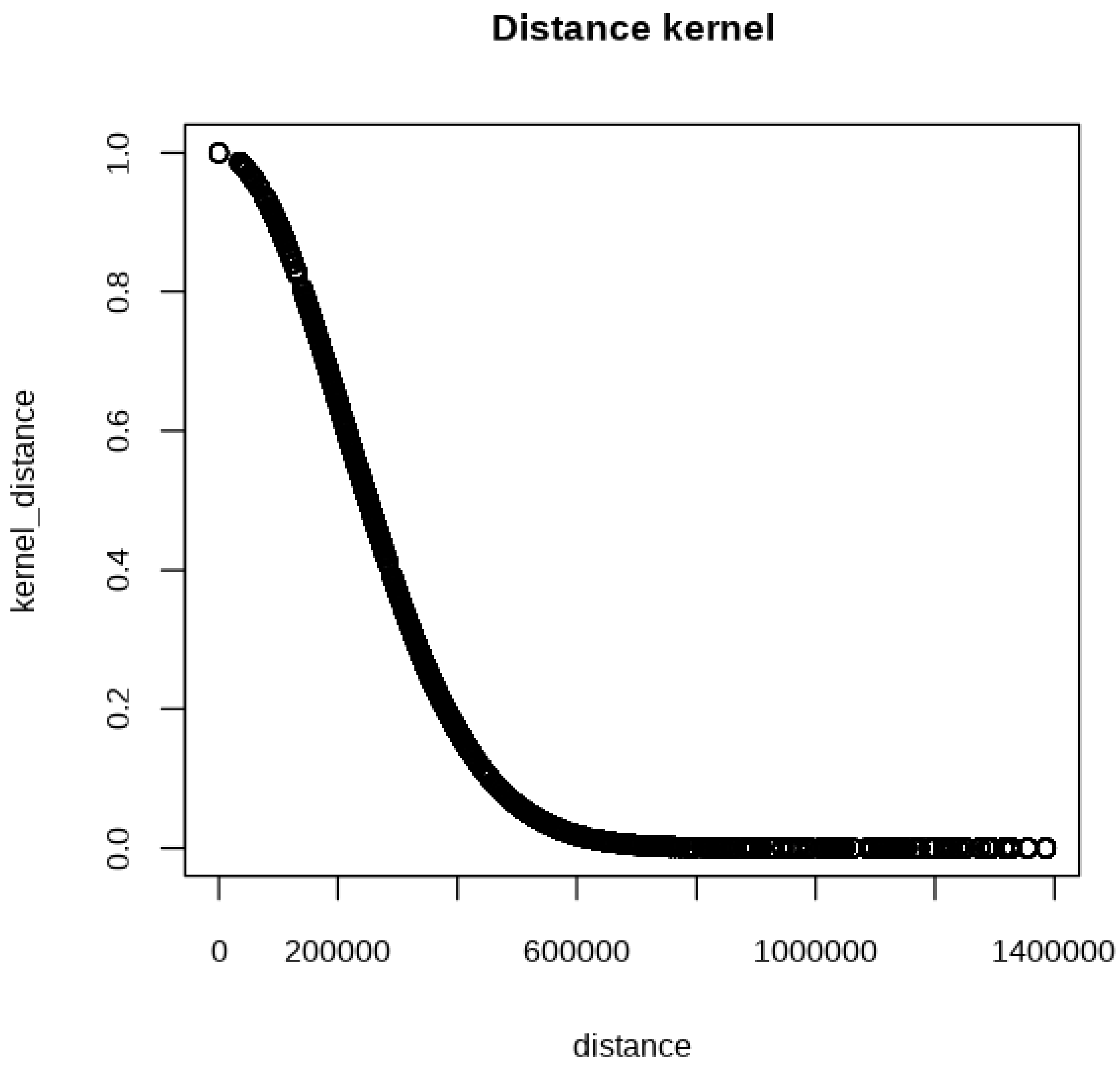
In [78]:
```r
#### DISTANCE

dist <- st[,c(1,2,4,5)]
stations <- unique(dist)
physical_dist <- distHaversine(p2, stations[,3:4])
stations$dist <- physical_dist
st_2  <- merge(stations,st,by="station_number")
distance <- st_2$dist
```

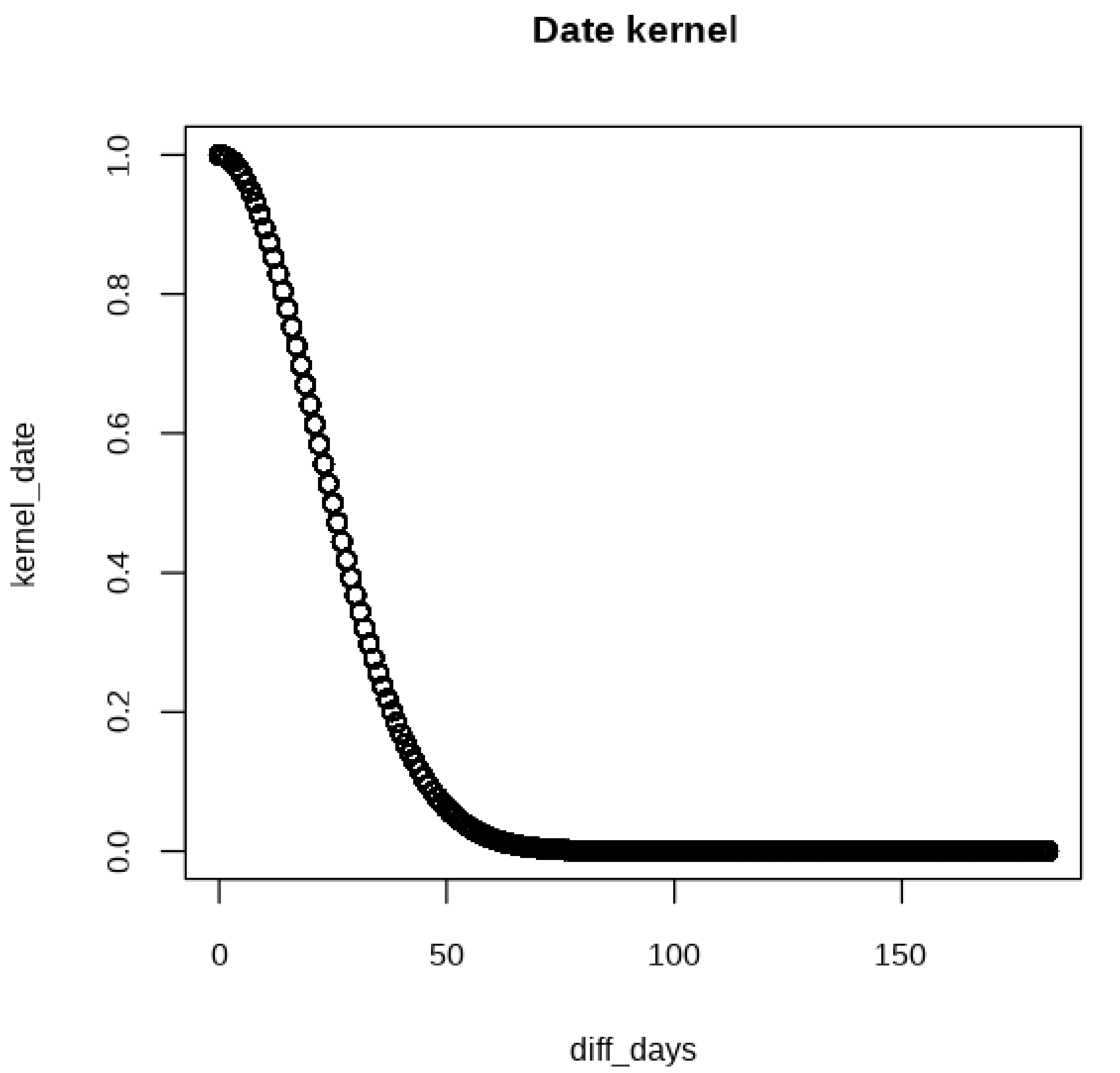
In [79]:
```r
### HOUR
times <- strptime(times, format = "%H:%M:%S")
st$time <- strptime(st$time, format = "%H:%M:%S")
time_diff <- data.frame(diff_4 =rep(0,nrow(st)), diff_6 = rep(0,nrow(st)), diff_8 =rep(0,nrow(st))
                        , diff_10 = rep(0,nrow(st)), diff_12 = rep(0,nrow(st)), diff_14 = rep(0,nrow(st))
                        , diff_16 = rep(0,nrow(st)), diff_18 = rep(0,nrow(st)), diff_20 =rep(0,nrow(st))
                        , diff_22 =rep(0,nrow(st)), diff_24 = rep(0,nrow(st)))
for (i in 1:ncol(time_diff)) {
  time_diff[,i] <- (as.numeric(abs(difftime(times[i], st$time, unit = "hours"))))
  time_diff[,i] <- ifelse(time_diff[,i]<12, time_diff[,i] , 24-time_diff[,i])
}
```

First we do some data processing. We compute the distance between the by using distHaversine. For the days we begin to set all dates to the same year to be able to count the exact days diffs it is. After that we use ifelse to be able to get the correct number of days between to dates. For the hour we compute the the difference between times for all the different timepoints. We use ifelse here as well.

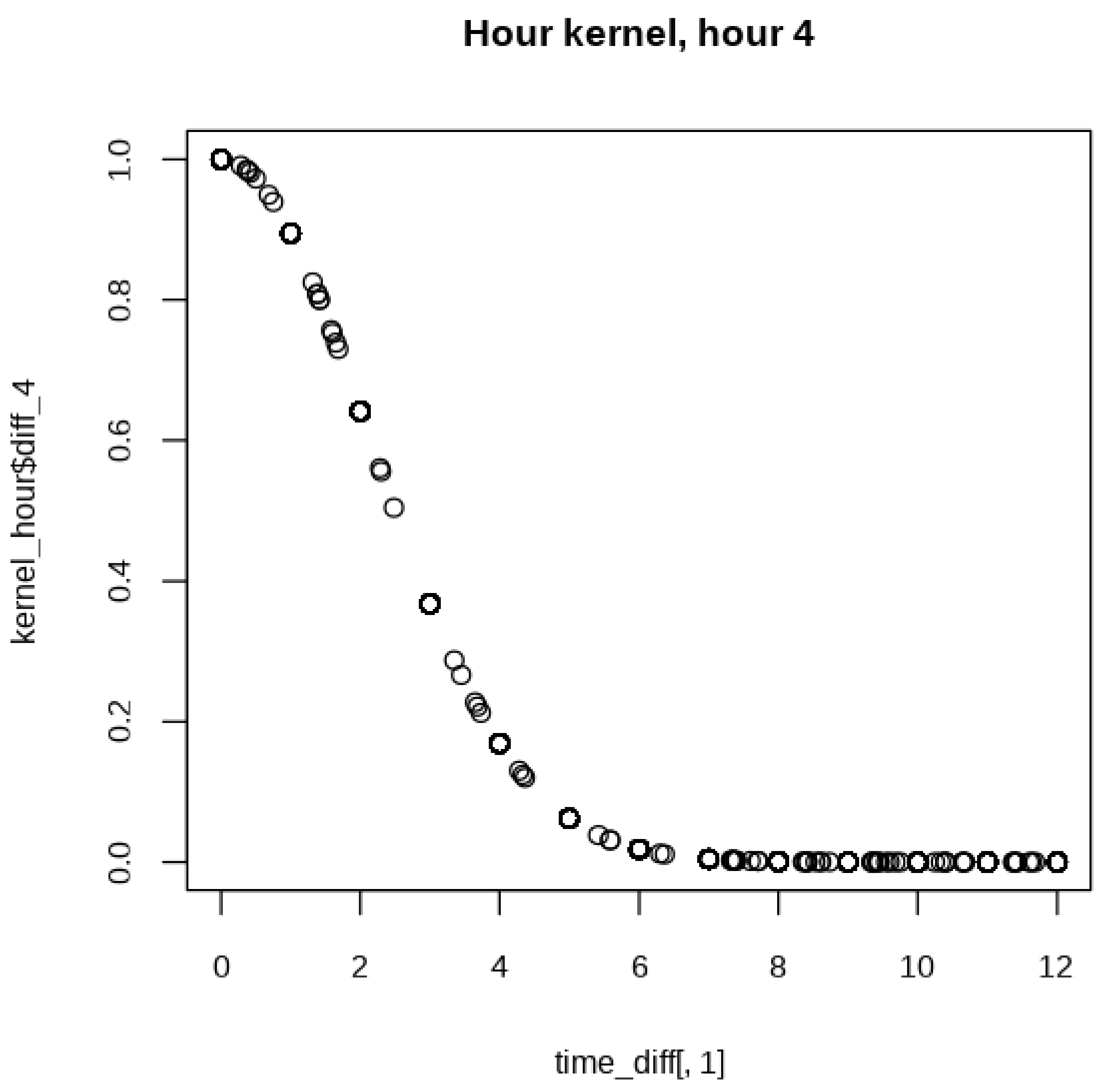
In [88]:
```
options(repr.plot.width=5, repr.plot.height=5)
#Distance smoothing
h_distance <- 300000
kernel_distance <- exp(-(distance/h_distance)^2)
plot(distance, kernel_distance, main = "Distance kernel")
```



In [89]:
```
#Day smoothing
h_date <- 30
kernel_date <- exp(-(diff_days/h_date)^2)
plot(diff_days, kernel_date, main = "Date kernel")
```



In [90]:
```
#Hour smoothing
h_time <- 3
kernel_hour <- exp(-(time_diff/h_time)^2)
plot(time_diff[,1], kernel_hour$diff_4, main = "Hour kernel, hour 4")
```



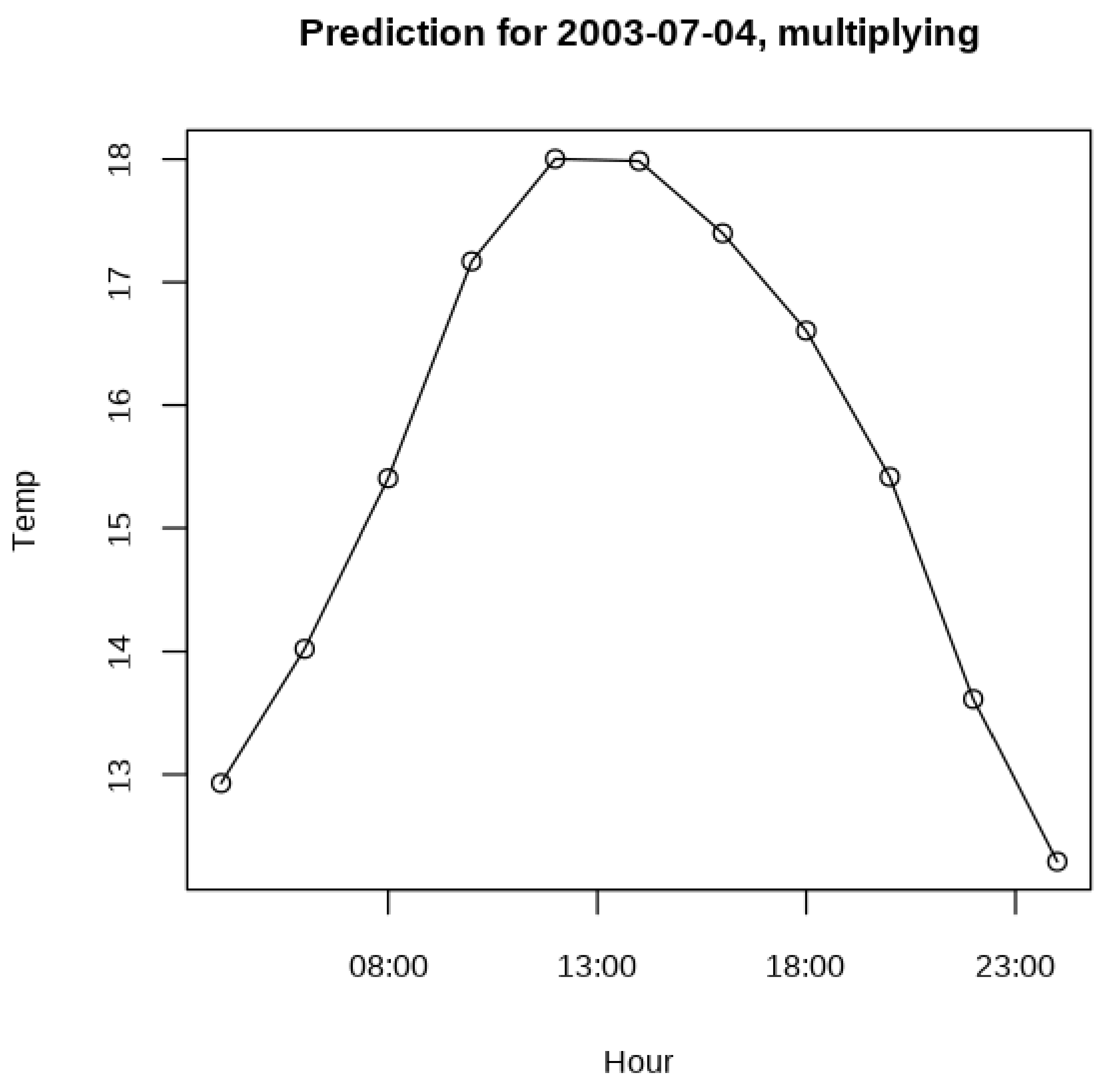
Here we can see the 3 different kernels for Distance, Date and Hour. For distance we choosed h 30000, for date h 30 and for hour h 3. This is very subjectively chosen but it feels reasonable when we have weather data.

Predicition, Multiplying Kernels.

In [92]:

```
#### MULTIPLYING KERNELS ####

for (i in 1:ncol(kernel_hour)){
 temp[i] <-  sum(kernel_distance*kernel_date*kernel_hour[,i]*st$air_temperature)/
   sum(kernel_distance*kernel_date*kernel_hour[,i])
}
mult_kernels <- data.frame(temp, times)
plot(mult_kernels$times, mult_kernels$temp, type = "o", main = "Prediction for 2003-07-04, multiplying", ylab = "Temp'
```

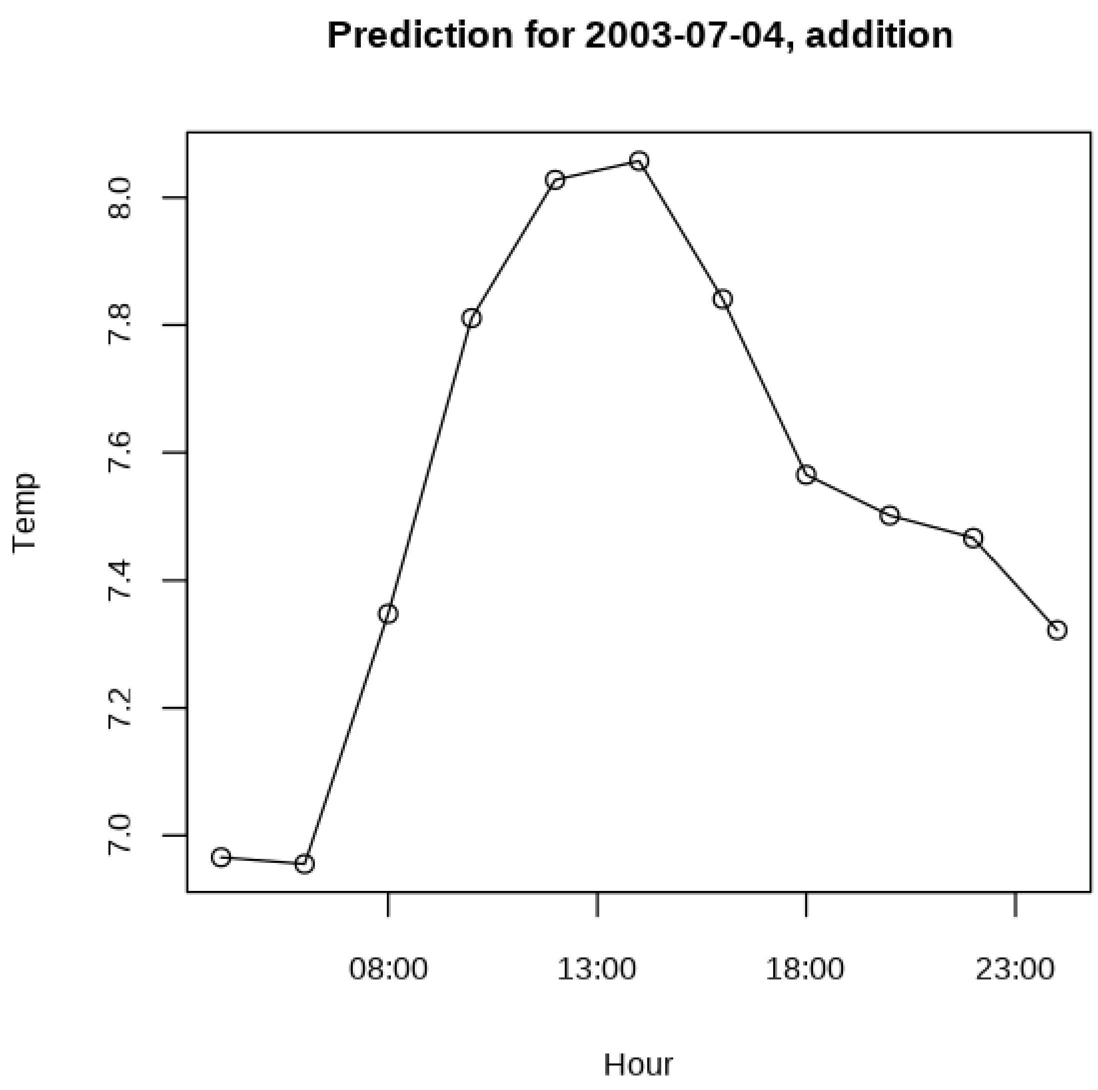


The plot above shows the predicted temperature for 4th July 2003 for different hours with multiplied kernels. We can see that the temperature looks reasonable for the model with lower temperatures and morning and night and high temperature around 12.00.

Predicition, Adding kernels.

In [91]:
```
#### ADDING KERNELS ####

for (i in 1:ncol(kernel_hour)){
  temp[i] <-  sum((kernel_distance+kernel_date+kernel_hour[,i])*st$air_temperature)/
    sum(kernel_distance+kernel_date+kernel_hour[,i])
}
add_kernels <- data.frame(temp, times)
plot(add_kernels$times, add_kernels$temp, type = "o", main = "Prediction for 2003-07-04, addition",
      ylab = "Temp", xlab = "Hour")
```



The plot above shows the predicted temperature for 4th July 2003 for different hours with summing kernels. We can see that the temperature looks not reasonable for the model with very low temperatures for the whole day and that it only differs one degree on the whole day.

Conclusion

We can see that the model with multiplying kernels are much better than the model with summing kernels. This may be due to when we add kernels together, a kernel with very small values get no impact at all and kernels with large values gets all the impact. If we compare that with when we multiplies kernels all kernels are still given an effect to the prediction because even if the value is small or big.

In [ ]: