lab3_block1_1

Zhixuan_Duan(zhidu838)

4/20/2020

Load packages

```
library(geosphere)
library(kernlab)
library(ggplot2)
library(caret)
```

Assignment 1. KERNEL METHODS

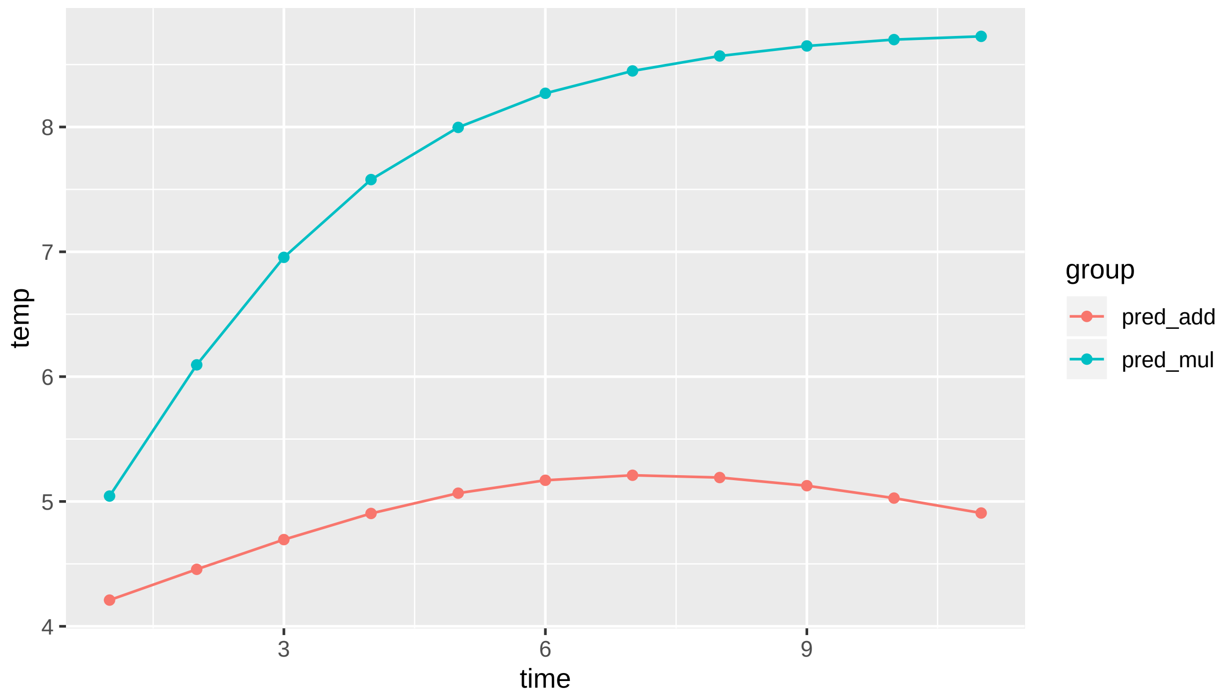
```
# import dataset
set.seed(1234567890)
stations <- read.csv("/Users/darin/Desktop/stations.csv",
                     fileEncoding="ISO-8859-1")
temps <- read.csv("/Users/darin/Desktop/temps50k.csv")
combined_data <- merge(stations,temps,by="station_number")
rm(stations, temps)

# build function
kernel_func <- function(a, b, date, h_distance, h_date, h_time){
  set.seed(123)
  df <- combined_data
  df <- df[,-3]
  df$old_date_time <- paste(df$date, df$time)

  # deal with time
  start <- as.POSIXct(date)
  interval <- 60*120
  end <- start + as.difftime(1, units="days")

  # predict date and time
  pred_date_time <- seq(from=start, to = end, by=interval)
  pred_date_time <- as.data.frame(pred_date_time[3:length(pred_date_time)])
  colnames(pred_date_time) <- "new_date_time"
  pred_date_time$new_date <- as.Date(pred_date_time$new_date_time)
  pred_date_time$new_time <- format(pred_date_time$new_date_time,"%H:%M:%S")
  pred_date_time$index <- rownames(pred_date_time)
  pred_date_time

  #date_time$index <- rownames(date_time)
  df_new <- merge.data.frame(df,pred_date_time,all=TRUE)
  rm(df, pred_date_time)
  df_new$a <- a
  df_new$b <- b
```

```r
  # compute distances
  # d_location
  df_new$d_location <- abs(distHaversine(p1 = df_new[,c('b', 'a')],
                                         p2 = df_new[,c("longitude", "latitude")]))
  # d_date
  df_new$d_date <- as.numeric(abs(difftime(df_new$new_date,
                                           df_new$old_date,
                                           units = 'days')))
  df_new$d_time <- as.numeric(abs(difftime(strptime(df_new$new_date_time,"%Y-%m-%d %H:%M:%S"),
                                           strptime(paste(df_new$new_date, df_new$time),"%Y-%m-%d %H:%M
                                                    units = c("hours"))))
  # compute kernel
  df_new$k_d_location <- exp(-(df_new$d_location/h_distance)^2)
  df_new$k_d_date <- exp(-(df_new$d_date/h_date)^2)
  df_new$k_d_time <- exp(-(df_new$d_time/h_time)^2)
  df_new$add <- df_new$k_d_location + df_new$k_d_date + df_new$k_d_time
  df_new$mul <- df_new$k_d_location * df_new$k_d_date * df_new$k_d_time
  df_new$add_num <- df_new$add * df_new$air_temperature
  df_new$mul_num <- df_new$mul * df_new$air_temperature
  index <- as.numeric(unique(df_new$index))
  results <- NULL

  # get the predict value
  for(i in index){
    temp <- df_new[df_new$index == i,]
    pred_add <- sum(temp$add_num)/sum(temp$add)
    pred_mul <- sum(temp$mul_num)/sum(temp$mul)
    pred <- cbind(pred_add, pred_mul, i)
    results <- rbind(pred,results)
  }
  temp1 <- as.data.frame(results)
  temp2 <- temp1[order(temp1$i),]
  temp3 <- temp2[,c(1,3)]
  temp3$group = 'pred_add'
  temp4 <- temp2[,c(2,3)]
  temp4$group = 'pred_mul'
  colnames(temp3) <- c('temp','time','group')
  colnames(temp4) <- c('temp','time','group')
  temp5 <- rbind(temp3,temp4)

  # plot
  ggplot(temp5, aes(x = time, y = temp, group = group)) +
    geom_line(aes(color = group)) +
    geom_point(aes(color = group))
}
```

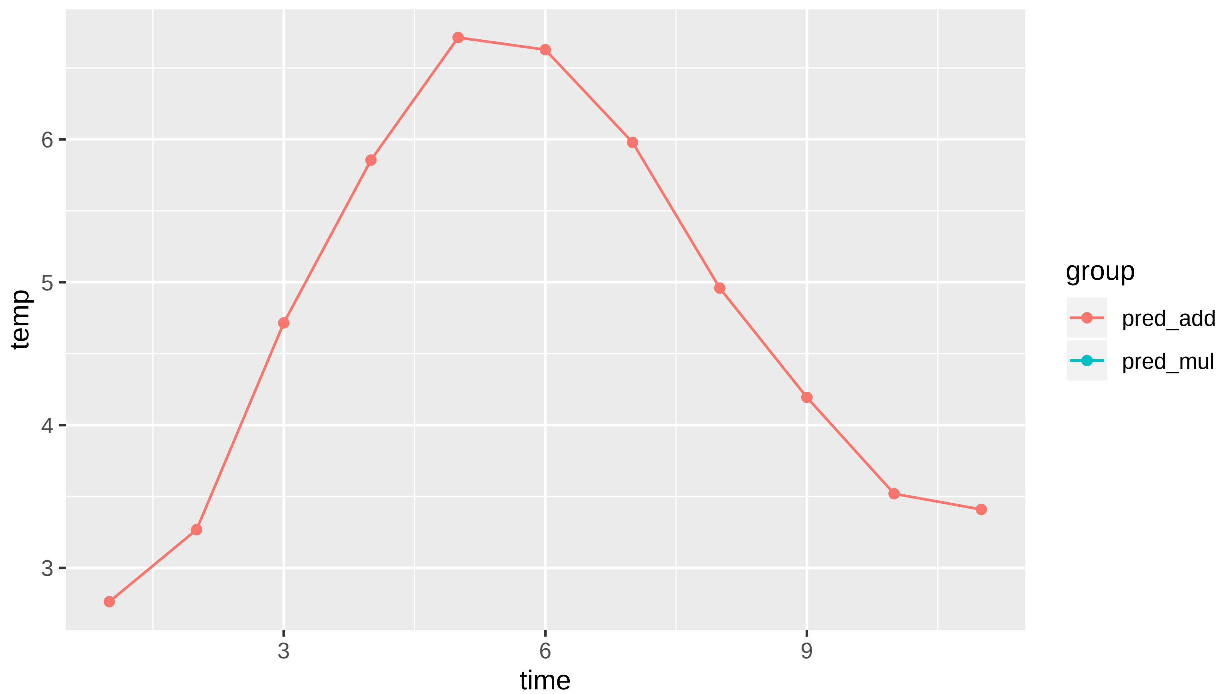
Analysis: Here we choose two sets of values to find better band widths. The first set are 1000, 100 and 10, and the second one are 100, 10 and 1.

```r
a <- 58.4274
b <- 14.826
date <- '2013-11-04'

h_distance <- 1000
```

```r
h_date <- 100
h_time <- 10

kernel_func(a, b, date, h_distance, h_date, h_time)
```

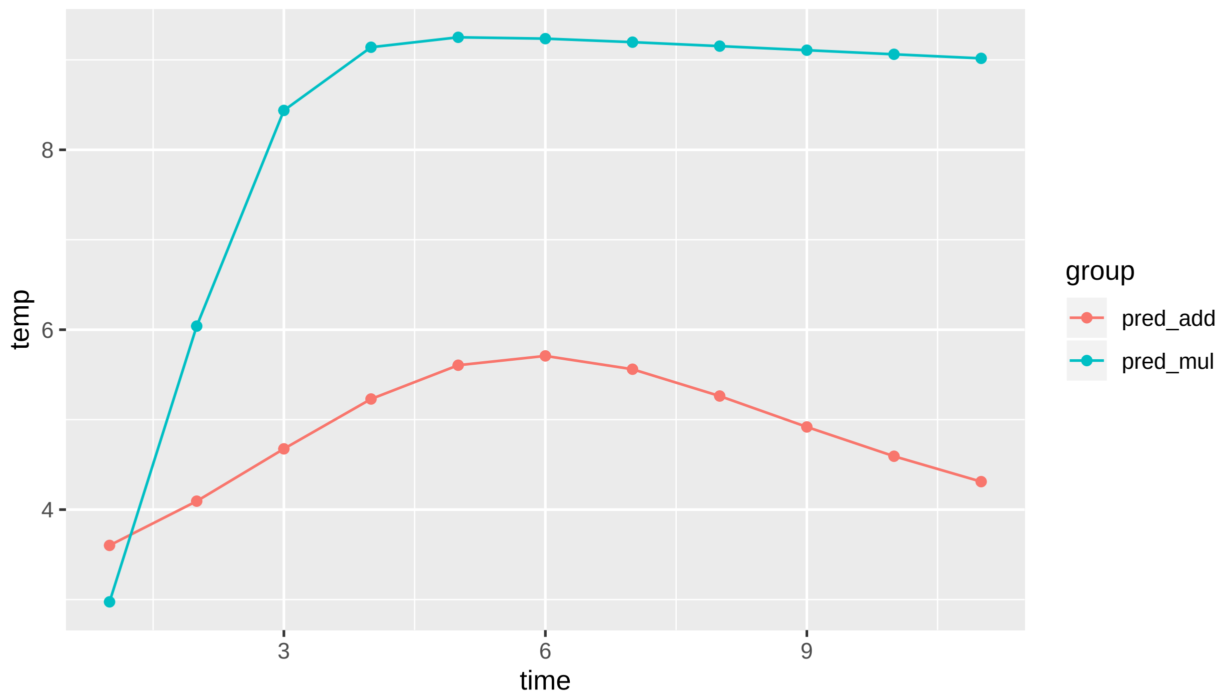



```r
h_distance <- 100
h_date <- 10
h_time <- 1

kernel_func(a, b, date, h_distance, h_date, h_time)
```



Analysis: From the outputs, we choose h_distance as 1000 considering the Sweden's land area. The h_date is 30, and the h_time is 6, since the temperature of the whole year is relatively smooth, but the temperature during the day may change significantly, these two values would give 'closer' values more weight. And here is the results.

```
h_distance <- 1000
h_date <- 30
h_time <- 6

kernel_func(a, b, date, h_distance, h_date, h_time)
```



Analysis: From the plot, we found that the cumulative model has greater volatility, while the cumulative model is gentler.

The reason for this phenomenon may be that the multiplicative model is more sensitive to outliers, resulting in a greater influence in the final model. The cumulative model will be relatively smooth,

Assignment 2. SUPPORT VECTOR MACHINES

In the beginning, we separate data to training, validation and test sets as 50/30/20. And we use training data to train models.

```
data(spam)
# separate data to training,validation and test
set.seed(123)

n <- nrow(spam)

index1 <- sample(1:n, floor(n*0.5))
train <- spam[index1,]

temp <- setdiff(1:n, index1)
index2 <- sample(temp, floor(n*0.3))
valid <- spam[index2,]

index3 <- setdiff(temp,index2)
test <- spam[index3,]

# train SVMs using different C values
model_0.5 <- ksvm(type~., data=train,kernel="rbfdot",
                   kpar=list(sigma=0.05), C=0.5)
model_1 <- ksvm(type~.,data=train, kernel="rbfdot",
                  kpar=list(sigma=0.05), C=1)
model_5 <- ksvm(type~.,data=train, kernel="rbfdot",
                  kpar=list(sigma=0.05), C=5)
```

Then we use validation dataset to evaluate these three models.

```
# confusion table
conf_model_0.5 <- table(valid[,58], predict(model_0.5,valid[,-58]))
names(dimnames(conf_model_0.5)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_0.5)
```

```
## Confusion Matrix and Statistics
##
##              Predicted Valid
## Actual Valid nonspam spam
##      nonspam     812   31
##      spam         87  450
##
##                Accuracy : 0.9145
##                  95% CI : (0.8985, 0.9287)
##     No Information Rate : 0.6514
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8167
##
##  Mcnemar's Test P-Value : 4.124e-07
##
##             Sensitivity : 0.9032
```

```
##             Specificity : 0.9356
##          Pos Pred Value : 0.9632
##          Neg Pred Value : 0.8380
##              Prevalence : 0.6514
##          Detection Rate : 0.5884
##    Detection Prevalence : 0.6109
##       Balanced Accuracy : 0.9194
##
##        'Positive' Class : nonspam
##
```

```
conf_model_1 <- table(valid[,58], predict(model_1,valid[,-58]))
names(dimnames(conf_model_1)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_1)
```

```
## Confusion Matrix and Statistics
##
##             Predicted Valid
## Actual Valid nonspam spam
##      nonspam     807   36
##      spam         74  463
##
##                Accuracy : 0.9203
##                  95% CI : (0.9047, 0.934)
##     No Information Rate : 0.6384
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8302
##
##  Mcnemar's Test P-Value : 0.000419
##
##             Sensitivity : 0.9160
##             Specificity : 0.9279
##          Pos Pred Value : 0.9573
##          Neg Pred Value : 0.8622
##              Prevalence : 0.6384
##          Detection Rate : 0.5848
##    Detection Prevalence : 0.6109
##       Balanced Accuracy : 0.9219
##
##        'Positive' Class : nonspam
##
```

```
conf_model_5 <- table(valid[,58], predict(model_5,valid[,-58]))
names(dimnames(conf_model_5)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_5)
```

```
## Confusion Matrix and Statistics
##
##             Predicted Valid
## Actual Valid nonspam spam
##      nonspam     803   40
##      spam         73  464
```

```
##
##                Accuracy : 0.9181
##                  95% CI : (0.9024, 0.932)
##     No Information Rate : 0.6348
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8258
##
##  Mcnemar's Test P-Value : 0.00261
##
##             Sensitivity : 0.9167
##             Specificity : 0.9206
##          Pos Pred Value : 0.9526
##          Neg Pred Value : 0.8641
##              Prevalence : 0.6348
##          Detection Rate : 0.5819
##    Detection Prevalence : 0.6109
##       Balanced Accuracy : 0.9187
##
##        'Positive' Class : nonspam
##
```

Analysis: From these three confusion matrix, we choose model_1 based on the accuracy. And we use test data to see the final output.

```
# choose the model_1
final <- ksvm(type~., data=test, kernel="rbfdot",
              kpar=list(sigma=0.05), C=1)
conf_final <- table(test[,58], predict(final, test[,-58]))
names(dimnames(conf_final)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_final)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test nonspam spam
##     nonspam     543   12
##     spam         27  339
##
##                Accuracy : 0.9577
##                  95% CI : (0.9426, 0.9697)
##     No Information Rate : 0.6189
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.911
##
##  Mcnemar's Test P-Value : 0.02497
##
##             Sensitivity : 0.9526
##             Specificity : 0.9658
##          Pos Pred Value : 0.9784
##          Neg Pred Value : 0.9262
##              Prevalence : 0.6189
```

```
##          Detection Rate : 0.5896
##    Detection Prevalence : 0.6026
##       Balanced Accuracy : 0.9592
##
##        'Positive' Class : nonspam
##
```

```r
# compute the generalization error, using the whole dataset
gen <- ksvm(type~., data=spam, kernel="rbfdot",
              kpar=list(sigma=0.05), C=1)
conf_gen <- table(spam[,58], predict(gen, spam[,-58]))
names(dimnames(conf_gen)) <- c("Actual Gen", "Predicted Gen")
confusionMatrix(conf_gen)
```

```
## Confusion Matrix and Statistics
##
##             Predicted Gen
## Actual Gen nonspam spam
##    nonspam    2727   61
##    spam        122 1691
##
##                Accuracy : 0.9602
##                  95% CI : (0.9542, 0.9657)
##     No Information Rate : 0.6192
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9162
##
##  Mcnemar's Test P-Value : 9.193e-06
##
##             Sensitivity : 0.9572
##             Specificity : 0.9652
##          Pos Pred Value : 0.9781
##          Neg Pred Value : 0.9327
##              Prevalence : 0.6192
##          Detection Rate : 0.5927
##    Detection Prevalence : 0.6060
##       Balanced Accuracy : 0.9612
##
##        'Positive' Class : nonspam
##
```

Analysis: The generalized model error of model_1 is 3.98% (Accuracy 96.02%).

The purpose of the 'C' parameter is to penalize large resuduals computed by model. The greater the C parameter, the more data would be contained in the model, leads to less bias, and higher variance.

Appendix

```
knitr::opts_chunk$set(echo = TRUE,
                      warning = FALSE,
                      message = FALSE,
                      fig.width = 7,
                      fig.height = 4,
                      fig.align = 'center')
library(geosphere)
library(kernlab)
library(ggplot2)
library(caret)

# import dataset
set.seed(1234567890)
stations <- read.csv("/Users/darin/Desktop/stations.csv",
                     fileEncoding="ISO-8859-1")
temps <- read.csv("/Users/darin/Desktop/temps50k.csv")
combined_data <- merge(stations,temps,by="station_number")
rm(stations, temps)

# build function
kernel_func <- function(a, b, date, h_distance, h_date, h_time){
  set.seed(123)
  df <- combined_data
  df <- df[,-3]
  df$old_date_time <- paste(df$date, df$time)

  # deal with time
  start <- as.POSIXct(date)
  interval <- 60*120
  end <- start + as.difftime(1, units="days")

  # predict date and time
  pred_date_time <- seq(from=start, to = end, by=interval)
  pred_date_time <- as.data.frame(pred_date_time[3:length(pred_date_time)])
  colnames(pred_date_time) <- "new_date_time"
  pred_date_time$new_date <- as.Date(pred_date_time$new_date_time)
  pred_date_time$new_time <- format(pred_date_time$new_date_time,"%H:%M:%S")
  pred_date_time$index <- rownames(pred_date_time)
  pred_date_time

  #date_time$index <- rownames(date_time)
  df_new <- merge.data.frame(df,pred_date_time,all=TRUE)
  rm(df, pred_date_time)
  df_new$a <- a
  df_new$b <- b

  # compute distances
  # d_location
  df_new$d_location <- abs(distHaversine(p1 = df_new[,c('b', 'a')],
                                         p2 = df_new[,c("longitude", "latitude")]))
  # d_date
```

```r
  df_new$d_date <- as.numeric(abs(difftime(df_new$new_date,
                                           df_new$old_date,
                                           units = 'days')))
  df_new$d_time <- as.numeric(abs(difftime(strptime(df_new$new_date_time,"%Y-%m-%d %H:%M:%S"),
                                           strptime(paste(df_new$new_date, df_new$time),"%Y-%m-%d %H:%M
                                                    units = c("hours"))))
  # compute kernel
  df_new$k_d_location <- exp(-(df_new$d_location/h_distance)^2)
  df_new$k_d_date <- exp(-(df_new$d_date/h_date)^2)
  df_new$k_d_time <- exp(-(df_new$d_time/h_time)^2)
  df_new$add <- df_new$k_d_location + df_new$k_d_date + df_new$k_d_time
  df_new$mul <- df_new$k_d_location * df_new$k_d_date * df_new$k_d_time
  df_new$add_num <- df_new$add * df_new$air_temperature
  df_new$mul_num <- df_new$mul * df_new$air_temperature
  index <- as.numeric(unique(df_new$index))
  results <- NULL

  # get the predict value
  for(i in index){
    temp <- df_new[df_new$index == i,]
    pred_add <- sum(temp$add_num)/sum(temp$add)
    pred_mul <- sum(temp$mul_num)/sum(temp$mul)
    pred <- cbind(pred_add, pred_mul, i)
    results <- rbind(pred,results)
  }
  temp1 <- as.data.frame(results)
  temp2 <- temp1[order(temp1$i),]
  temp3 <- temp2[,c(1,3)]
  temp3$group = 'pred_add'
  temp4 <- temp2[,c(2,3)]
  temp4$group = 'pred_mul'
  colnames(temp3) <- c('temp','time','group')
  colnames(temp4) <- c('temp','time','group')
  temp5 <- rbind(temp3,temp4)

  # plot
  ggplot(temp5, aes(x = time, y = temp, group = group)) +
    geom_line(aes(color = group)) +
    geom_point(aes(color = group))
}
a <- 58.4274
b <- 14.826
date <- '2013-11-04'

h_distance <- 1000
h_date <- 100
h_time <- 10

kernel_func(a, b, date, h_distance, h_date, h_time)

h_distance <- 100
h_date <- 10
h_time <- 1
```

```r
kernel_func(a, b, date, h_distance, h_date, h_time)
h_distance <- 1000
h_date <- 30
h_time <- 6

kernel_func(a, b, date, h_distance, h_date, h_time)
data(spam)
# separate data to training,validation and test
set.seed(123)

n <- nrow(spam)

index1 <- sample(1:n, floor(n*0.5))
train <- spam[index1,]

temp <- setdiff(1:n, index1)
index2 <- sample(temp, floor(n*0.3))
valid <- spam[index2,]

index3 <- setdiff(temp,index2)
test <- spam[index3,]

# train SVMs using different C values
model_0.5 <- ksvm(type~., data=train,kernel="rbfdot",
                   kpar=list(sigma=0.05), C=0.5)
model_1 <- ksvm(type~.,data=train, kernel="rbfdot",
                  kpar=list(sigma=0.05), C=1)
model_5 <- ksvm(type~.,data=train, kernel="rbfdot",
                  kpar=list(sigma=0.05), C=5)

# confusion table
conf_model_0.5 <- table(valid[,58], predict(model_0.5,valid[,-58]))
names(dimnames(conf_model_0.5)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_0.5)

conf_model_1 <- table(valid[,58], predict(model_1,valid[,-58]))
names(dimnames(conf_model_1)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_1)

conf_model_5 <- table(valid[,58], predict(model_5,valid[,-58]))
names(dimnames(conf_model_5)) <- c("Actual Valid", "Predicted Valid")
confusionMatrix(conf_model_5)
# choose the model_1
final <- ksvm(type~., data=test, kernel="rbfdot",
              kpar=list(sigma=0.05), C=1)
conf_final <- table(test[,58], predict(final, test[,-58]))
names(dimnames(conf_final)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_final)
# compute the generalization error, using the whole dataset
gen <- ksvm(type~., data=spam, kernel="rbfdot",
              kpar=list(sigma=0.05), C=1)
conf_gen <- table(spam[,58], predict(gen, spam[,-58]))
names(dimnames(conf_gen)) <- c("Actual Gen", "Predicted Gen")
```

```
confusionMatrix(conf_gen)
```

In the kernel model exercise, your code looks good but the experiments do not. You don't have to compare two sets of bandwidths. You have to select one based on intuition, i.e. plot the kernel values as a function of the bandwidth and select a bandwidth that gives almost zero kernel value to training points that you consider irrelevant (e.g. because they are too far again in space or time). Moreover, you need to improve your explanations. These sentences don't make sense to me:

"Analysis: From the plot, we found that the cumulative model has greater volatility, while the cumulative

model is gentler.

The reason for this phenomenon may be that the multiplicative model is more sensitive to outliers, resulting

in a greater influence in the final model. The cumulative model will be relatively smooth,"

In the SVM exercise, you compute the generalization error using either the test data for both training and testing or using the whole data for training and test. Both are wrong. You should use train+validation for training and test in the test data. Moreover, your answer for the question about C is wrong.