PSET 4

Boseong Yun

3/8/2021

Question 1

a. Load the data as a corpus. For doing so, use the following command (leveraging the tm package):

```
# Loading the data as a corpus
texts <- file.path("/Users/boseongyun/Desktop/ml/PSET 4/SimpleText_auto") # add local file path!
docs <- VCorpus(DirSource(texts))
```

b. Clean the data. This implies transforming all characters to lowercase and removing stop words, punctuation, and any other words that will not generate meaningful content for identifying the topics. Think about words that are likely common in academic papers (e.g., table, figure, results). Also think about combining forms of the same word (e.g., genes and gene). Be sure to justify your decisions.

- **Answer: I have trasnsformed all characters to lowercase and removed stop words, punctuation, whitespace, and numbers. I have removed whitespace and numbers because they contribute little to understanding the topics of academic papers. More specifically, it is hard to determine what whitespace and numbers suggest without access to detailed contextual information. I have also removed words that frequently appear in academic papers to improve the model's power in detecting relevant topics. Finally, I combined frequent forms of the same words that appear across the academic papers. I have found these words by running the analysis first and detecting their common appearances. I have not used the StemDocument function that automatically deals with similar words because it creates words such as "studi" that might be misleading. Since I have less information about the entire Corpus and many terminologies in each field, I thought it was prudent to not use the StemDocument function.**

```
# Cleaning the data
# Source: https://stackoverflow.com/questions/48545106/using-tm-package-in-r-to-clean
-the-columns-in-dataframe

# Cleaning 1: Transforming all characters to lower case
docs2 <- tm_map(docs, content_transformer(tolower))

# Cleaning 2: Removing all stop words in english
docs2 <- tm_map(docs2, removeWords, stopwords('english'))

# Celaning 3: Remove Selected Stop Words for acadmeic journals
docs2 <- tm_map(docs2, removeWords, c("figure", "figures", "table", "tables",
                                      "result", "results", "fig", "analysis",
                                      "analyses", "section", "study", "studies",
                                      "can", "also"))

# Cleaning 4: Removing Punctuations
docs2 <- tm_map(docs2, removePunctuation)

# Cleaning 5: Removing White Spaces
docs2 <- tm_map(docs2, stripWhitespace)

# Cleaning 6: Removing Numbers
docs2 <- tm_map(docs2, removeNumbers)

# Cleaning 7: Removing Stem Documetnts Creates too much information loss
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "gene|genes", replacement
= "genes")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "cell|cells", replacement
= "cells")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "level|levels", replaceme
nt = "levels")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "sample|samples", replace
ment = "samples")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "value|values", replaceme
nt = "values")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "cluster|clusters", repla
cement = "clusters")
docs2 <- tm_map(docs2, content_transformer(gsub), pattern = "node|nodes", replacement
= "nodes")

# Saving the results
result <- unlist(as.list(docs2))

# Saving the results as a dataframe
result_df <- result %>%
  bind_rows() %>%
  pivot_longer(1:ncol(.))
```

c. Present the 50 most frequently used words in the corpus in an informative way. This can include a table of results or a word cloud.

```
# Presenting the 50 most frequently used words in the corpus
result_df %>%
  unnest_tokens(word, value, token = "words") %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  head(n = 50) %>%
  set_names("Word", "Count") %>%
  knitr::kable(
    caption = "The 50 Most Frequently Words Used in the Corpus"
  )
```

The 50 Most Frequently Words Used in the Corpus

Word	Count
al	3759
model	1442
data	1379
cells	1305
values	1289
time	1066
levels	978
flow	893
energy	831
set	759
system	758
soil	753
due	748
surface	744
nodes	704
lower	696
observed	692
samples	681
temperature	665
based	651
field	651
clusters	636
wind	624
shown	615

Word	Count
low	594
conditions	558
similar	554
region	537
genes	536
found	535
size	530
increase	516
velocity	499
compared	483
power	456
electron	454
rate	454
species	453
approach	450
network	447
function	443
control	441
density	441
significant	440
particles	426
performance	425
algorithm	424
increased	421
ratio	420
measured	415

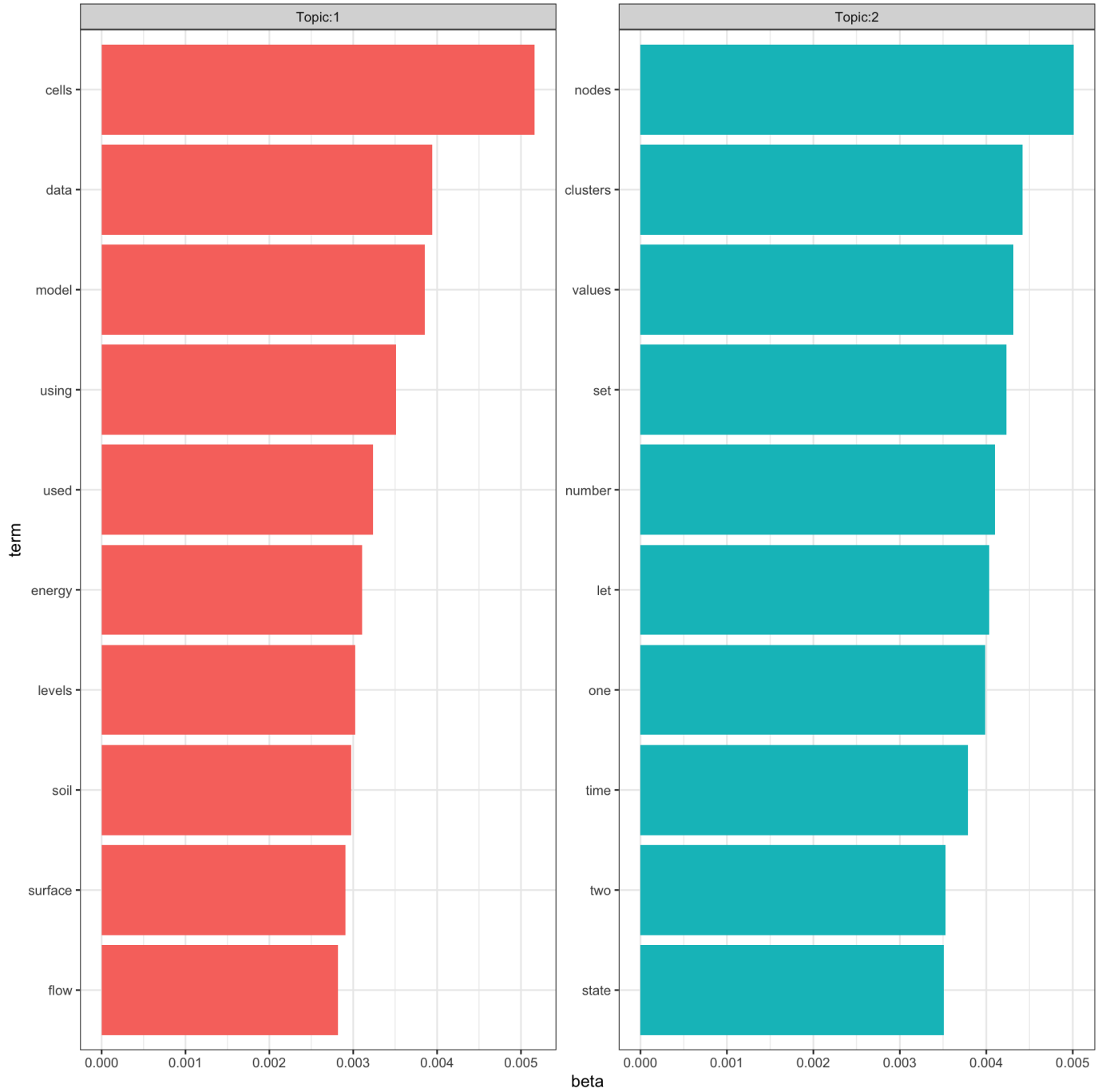
d. Fit a topic model on the corpus setting k equal to 2, 3, 5, 8, and 10. Present the topics for each value of k and interpret the topics. In your opinion, which of the selected values of k yield the most meaningful coherence for each topic?

- **Answer: The topics seem to start get more coherent after k = 5. For instance, when k = 3, it is hard to understand what topic2 and topic3 precisely capture. Althought it might be possible to infer that the topic is related to science, there does not seem to be extra information that indicates what kinds of science each topic represents. At k=5, however, it is easy to discern that topic2 seems to discuss biology and topic5 seems to discuss energy and thus provide some useful information. At K = 10, it is even possible to figure out the exact field of the study in the**

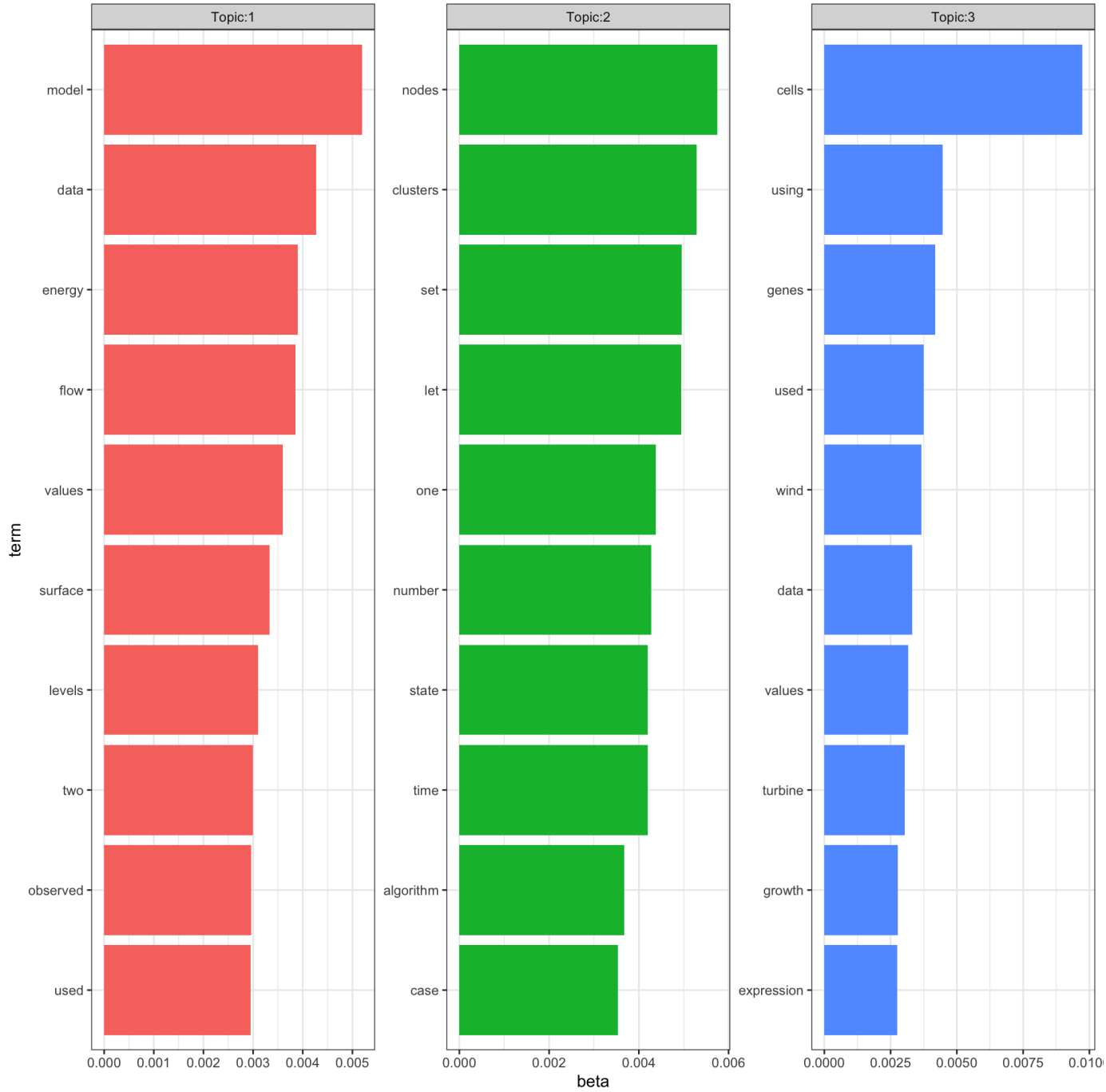
field. For instace, TOPIC 10 does not just talk about biology but human genes with every word relevant to the field! Thus, I believe that k = 10 yield the most meaningful coherenece for each topic!

```
# Transforming the corpus into a dtm object for LDA analysis
dtm <- DocumentTermMatrix(docs2)

# Creating a customized function that returns the
topic_analyze <- function(mod) {

  # Creating a dataframe
  mod_df <- mod %>%
    tidy(matrix = "beta") %>%
    group_by(topic) %>%
    top_n(10, beta) %>%
    ungroup() %>%
    arrange(topic, -beta)

  # Creating a ggplot aesthetic
  plot <- mod_df %>%
    mutate(term = reorder_within(term, beta, topic)) %>%
    mutate(
    topic = paste0("Topic:", topic),
    num = parse_number(topic),
    topic = reorder(topic, num)
     ) %>%
    ggplot(aes(x = term, y = beta, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    scale_x_reordered() +
    facet_wrap(~topic, scales = "free") +
    coord_flip()

  # return
  return(plot)
  }

# Creating a for loop

# Saving the output list
output <- list()

# Run the for loop
for(i in c(2, 3, 5, 8, 10)) {

  # Specifying the model
  mod_lda <- LDA(dtm,
                 k = i,
                 control = list(seed = 332))

  # Saving the output to the output path
  output[[i]] <- topic_analyze(mod_lda)

  }
```

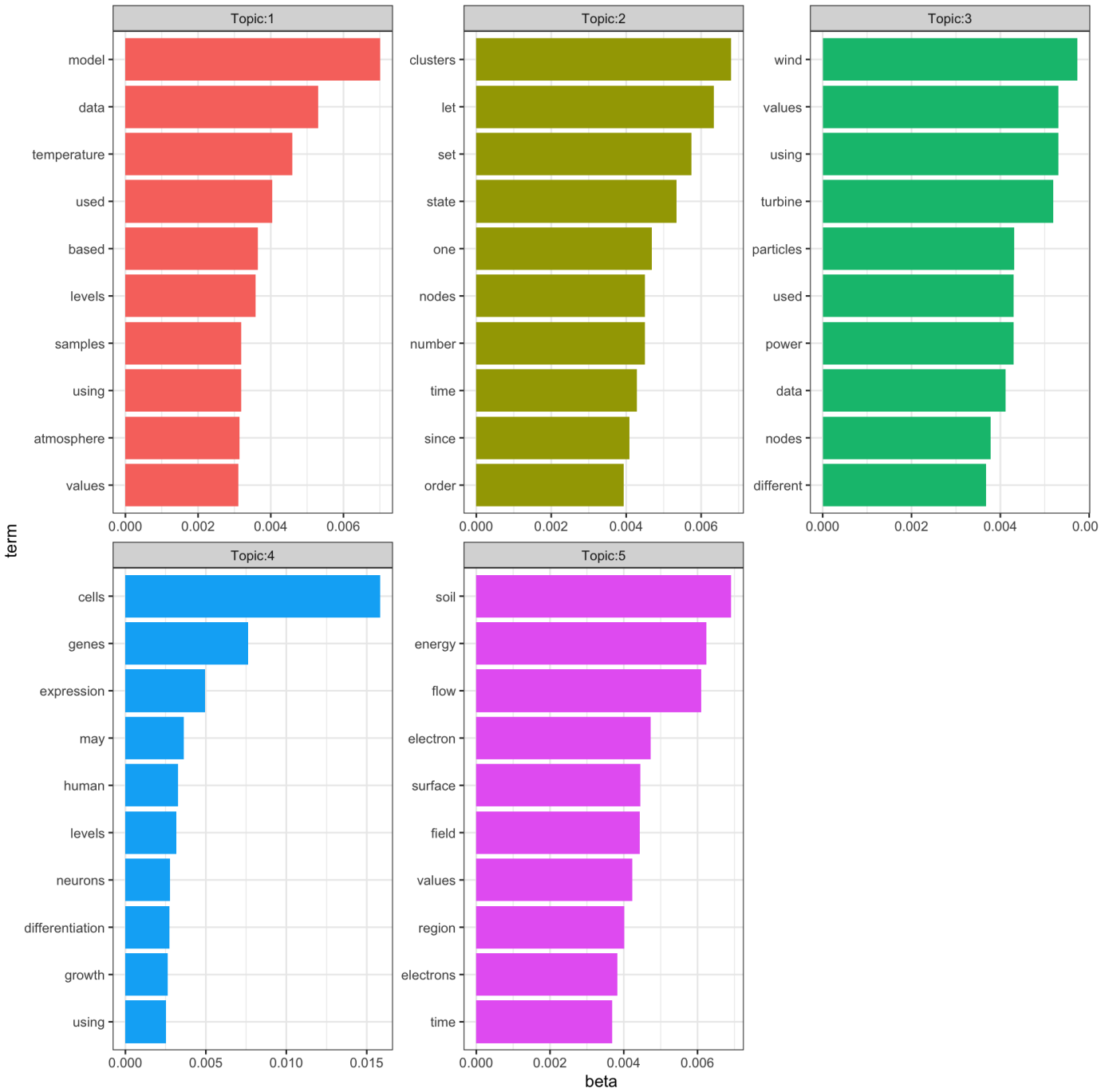
```
# Displaying the output
output[[2]]
```


Topic:1
Topic:2
cells
data
model
using
used
energy
levels
soil
surface
flow
nodes
clusters
values
set
number
let
one
time
two
state
term
beta
0.000
0.001
0.002
0.003
0.004
0.005

```
output[[3]]
```


Topic:1
Topic:2
Topic:3
term
model
data
energy
flow
values
surface
levels
two
observed
used
nodes
clusters
set
let
one
number
state
time
algorithm
case
cells
using
genes
used
wind
data
values
turbine
growth
expression
0.000
0.001
0.002
0.003
0.004
0.005
0.000
0.002
0.004
0.006
0.0000
0.0025
0.0050
0.0075
beta

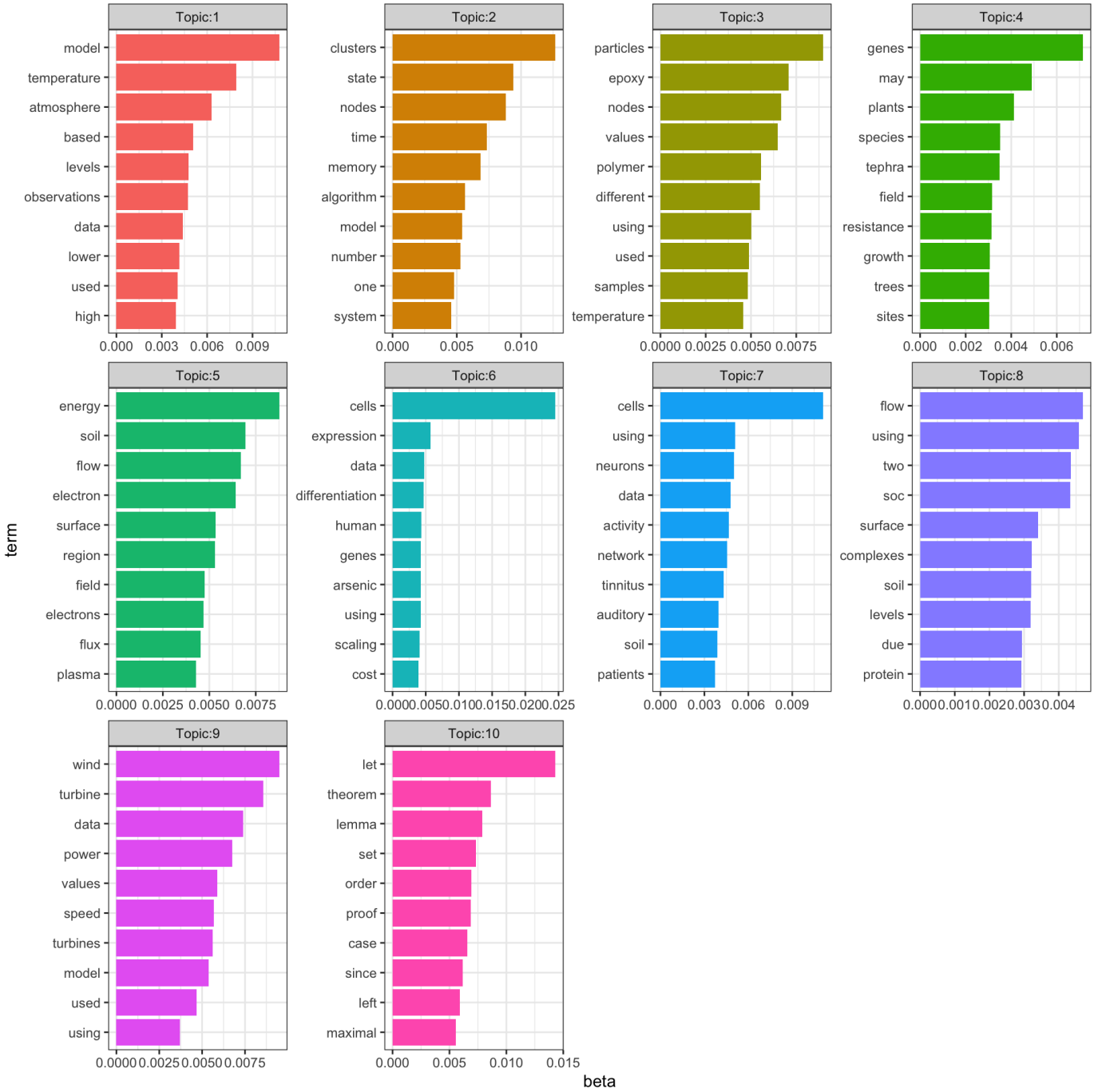
```
output[[5]]
```


Topic:1
model
data
temperature
used
based
levels
samples
using
atmosphere
values
Topic:2
clusters
let
set
state
one
nodes
number
time
since
order
Topic:3
wind
values
using
turbine
particles
used
power
data
nodes
different
Topic:4
cells
genes
expression
may
human
levels
neurons
differentiation
growth
using
Topic:5
soil
energy
flow
electron
surface
field
values
region
electrons
time
term
beta
0.000
0.002
0.004
0.006
0.005
0.010
0.015

```
output[[8]]
```


Topic:1
clusters
model
temperature
data
used
based
atmosphere
coherent
levels
observations
0.0000 0.0025 0.0050 0.0075 0.0100
Topic:2
let
state
nodes
theorem
set
since
memory
one
lemma
number
0.0000 0.0025 0.0050 0.0075
Topic:3
wind
turbine
power
data
particles
used
model
values
using
turbines
0.000 0.002 0.004 0.006
Topic:4
genes
may
plants
species
tephra
resistance
trees
growth
sites
selection
0.000 0.002 0.004 0.006
Topic:5
energy
flow
soil
surface
electron
region
field
time
model
values
0.000 0.002 0.004 0.006
Topic:6
cells
values
data
expression
arsenic
genes
scaling
differentiation
sox
algorithm
0.000 0.005 0.010 0.015
Topic:7
cells
using
data
neurons
activity
network
used
culture
tinnitus
patients
0.000 0.005 0.010
Topic:8
levels
flow
two
soc
values
using
soil
one
complexes
plate
0.000 0.001 0.002 0.003 0.004 0.005
term
beta

```
output[[10]]
```



e. Optimize the hyperparameters of the LDA model using 10-fold cross-validation. Present the topics from the best model and explain your results. Hint: The topicmodels package may be helpful. We have seen that in p = 2 dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X1 + \beta_2 X2 = 0$. We now investigate a non-linear decision boundary.

- **The result of optimizing the LDA model using 10-fold cross-validation shows that the optinmal number of topics is between 50 and 100. There is a noticeble flattening out of the cross-validated perplexity as the number of topic approaches 100. While it might be idea to extend the candidate topics to larger number of topics such as 200 or 500, it is computationally prohibitive to run the code (using my laptop). From the given graph, the model is likely to perform worse afterwards. Thus, it is reasonable to choose the topic number between 50 and 100. Although not required in this problem, these results are in consistent with the results from the *ldaunting* package that finds the optimal number of topics as 50. While 100 is also a good candidate, the computational costs prevent me from choosing 100 as the candidate. With both evidence showing 50 as an ideal number of the topics, I proceed my analysis with 50 topics**

- **The topics show strong coherency when the number of topics is set to 50. In topic 4, for instance, it is possible to infer that the topic is highly related to trees and plants. In topic 14, it is possible to infer that the topic is about mathetmatics where there are many terminologies such as maximal, fractional, and lemma. Other topics also show simliar level of details. Compared to (d) where the topic was set to 10, setting the number of topics to 50 vastly increased the specific details for every topic. As we increase the number of topics, we identify more fine-graind topics that help us map the main themes or topics for OA-STM-Corpus.**

Notes: I used the following resources to answer this question

- https://rpubs.com/MNidhi/NumberoftopicsLDA (https://rpubs.com/MNidhi/NumberoftopicsLDA)
- http://freerangestats.info/blog/2017/01/05/topic-model-cv (http://freerangestats.info/blog/2017/01/05/topic-model-cv)
- https://slcladal.github.io/topicmodels.html (https://slcladal.github.io/topicmodels.html)
- https://m-clark.github.io/text-analysis-with-R/topic-modeling.html (https://m-clark.github.io/text-analysis-with-R/topic-modeling.html)
- https://stackoverflow.com/questions/21355156/topic-models-cross-validation-with-loglikelihood-or-perplexity (https://stackoverflow.com/questions/21355156/topic-models-cross-validation-with-loglikelihood-or-perplexity)

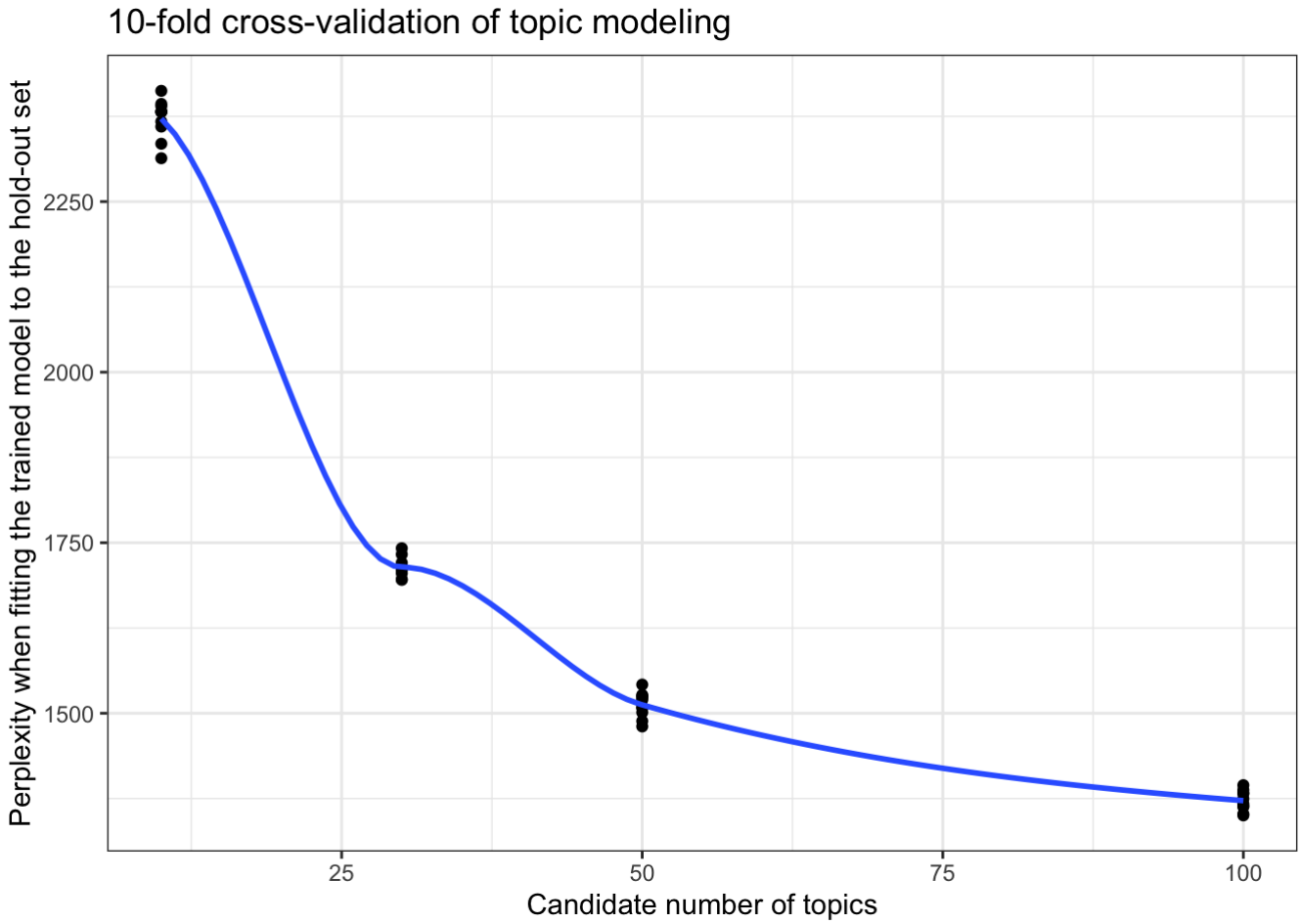
```
#----------------5-fold cross-validation, different numbers of topics----------------
parallel:::setDefaultClusterOptions(setup_strategy = "sequential")
cluster <- makeCluster(detectCores(logical = TRUE) - 1) # leave one CPU spare...
registerDoParallel(cluster)

clusterEvalQ(cluster, {
   library(topicmodels)
})
```

```
## [[1]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[2]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[3]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[4]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[5]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[6]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[7]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[8]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[9]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[10]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[11]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
```

```
# Setting the paramters
data <- dtm
folds <- 10
splitfolds <- sample(1:folds, replace = TRUE)
candidate_k <- c(10, 30, 50, 100) # candidates for how many topics
burnin <- 100
iter <- 1000
keep <- 50


clusterExport(cluster, c("data", "burnin", "iter", "keep", "splitfolds", "folds", "ca
ndidate_k"))

# we parallelize by the different number of topics.  A processor is allocated a value
# of k, and does the cross-validation serially.  This is because it is assumed therej
# are more candidate values of k than there are cross-validation folds, hence it
# will be more efficient to parallelise


# Source: http://freerangestats.info/blog/2017/01/05/topic-model-cv

system.time({
results <- foreach(j = 1:length(candidate_k), .combine = rbind) %dopar%{
   k <- candidate_k[j]
   results_1k <- matrix(0, nrow = folds, ncol = 2)
   colnames(results_1k) <- c("k", "perplexity")
   for(i in 1:folds){
      train_set <- data[splitfolds != i , ]
      valid_set <- data[splitfolds == i, ]

      fitted <- LDA(dtm, k = k, method = "Gibbs",
                    control = list(burnin = burnin, iter = iter, keep = keep))
      results_1k[i,] <- c(k, perplexity(fitted, newdata = dtm))
   }
   return(results_1k)
}
})
```

```
##     user   system  elapsed
##    0.194    0.041 4430.809
```

```
stopCluster(cluster)

# Saving the results
results_df <- as_tibble(results)

# Creating a ggplot
ggplot(results_df, aes(x = k, y = perplexity)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  ggtitle("10-fold cross-validation of topic modeling") +
  labs(
    x = "Candidate number of topics",
    y = "Perplexity when fitting the trained model to the hold-out set")
```




```
##plot the metrics to get number of topics
    parallel:::setDefaultClusterOptions(setup_strategy = "sequential")
    cluster <- makeCluster(detectCores(logical = TRUE) - 1) # leave one CPU spare...
    registerDoParallel(cluster)

    clusterEvalQ(cluster, {
       library(topicmodels)
    })
```

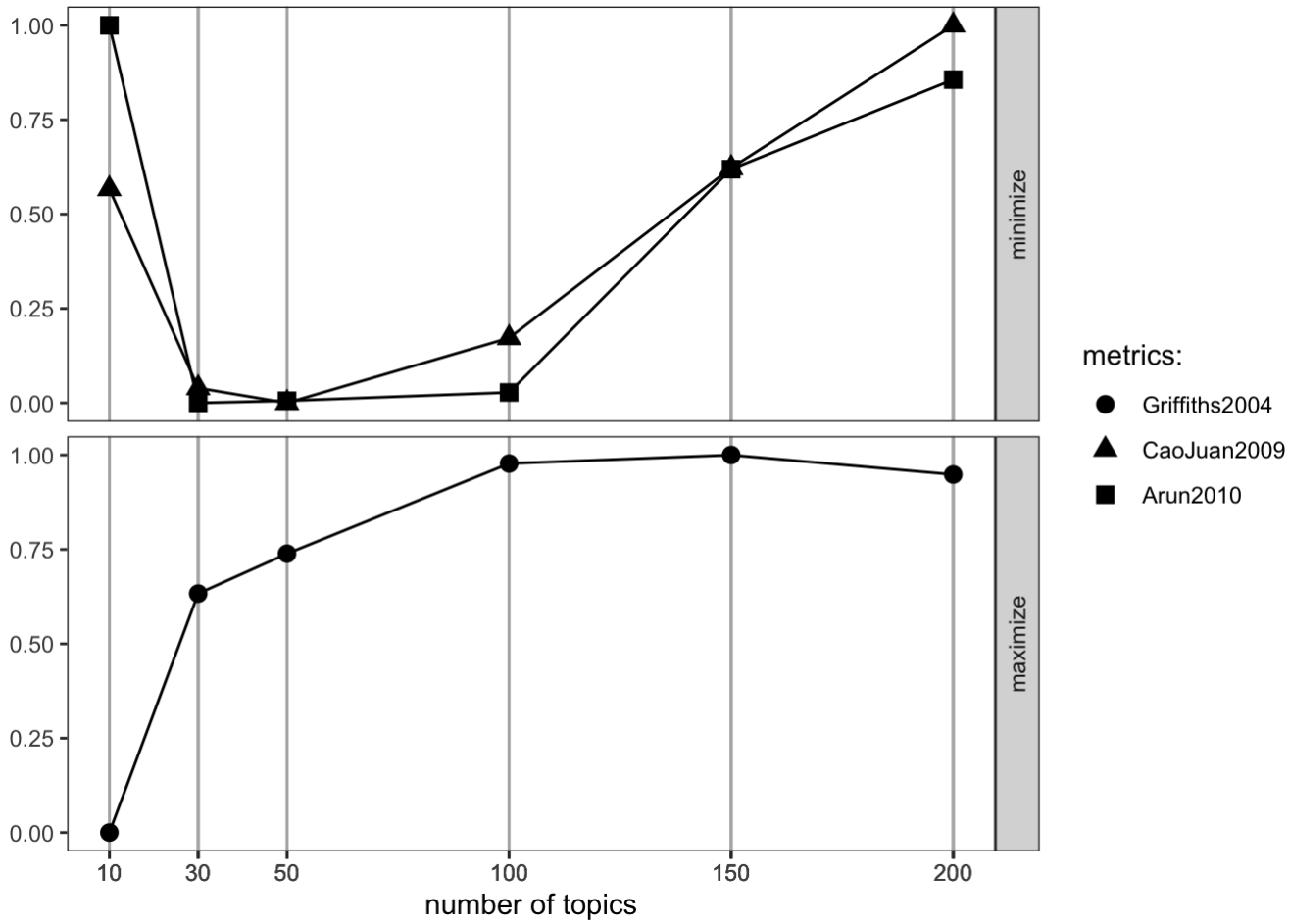
```
## [[1]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[2]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[3]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[4]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[5]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[6]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[7]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[8]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[9]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[10]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
##
## [[11]]
## [1] "topicmodels" "stats"       "graphics"    "grDevices"   "utils"
## [6] "datasets"    "methods"     "base"
```

```
    system.time({
      tunes <- FindTopicsNumber(
        dtm = dtm,
        topics = c(10, 30, 50, 100, 150, 200),
        metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010"),
        method = "Gibbs",
        control = list(seed = 12345),
        mc.cores = 4L,
        verbose = TRUE
      )
    })
```

```
## fit models... done.
## calculate metrics:
##   Griffiths2004... done.
##   CaoJuan2009... done.
##   Arun2010... done.
```

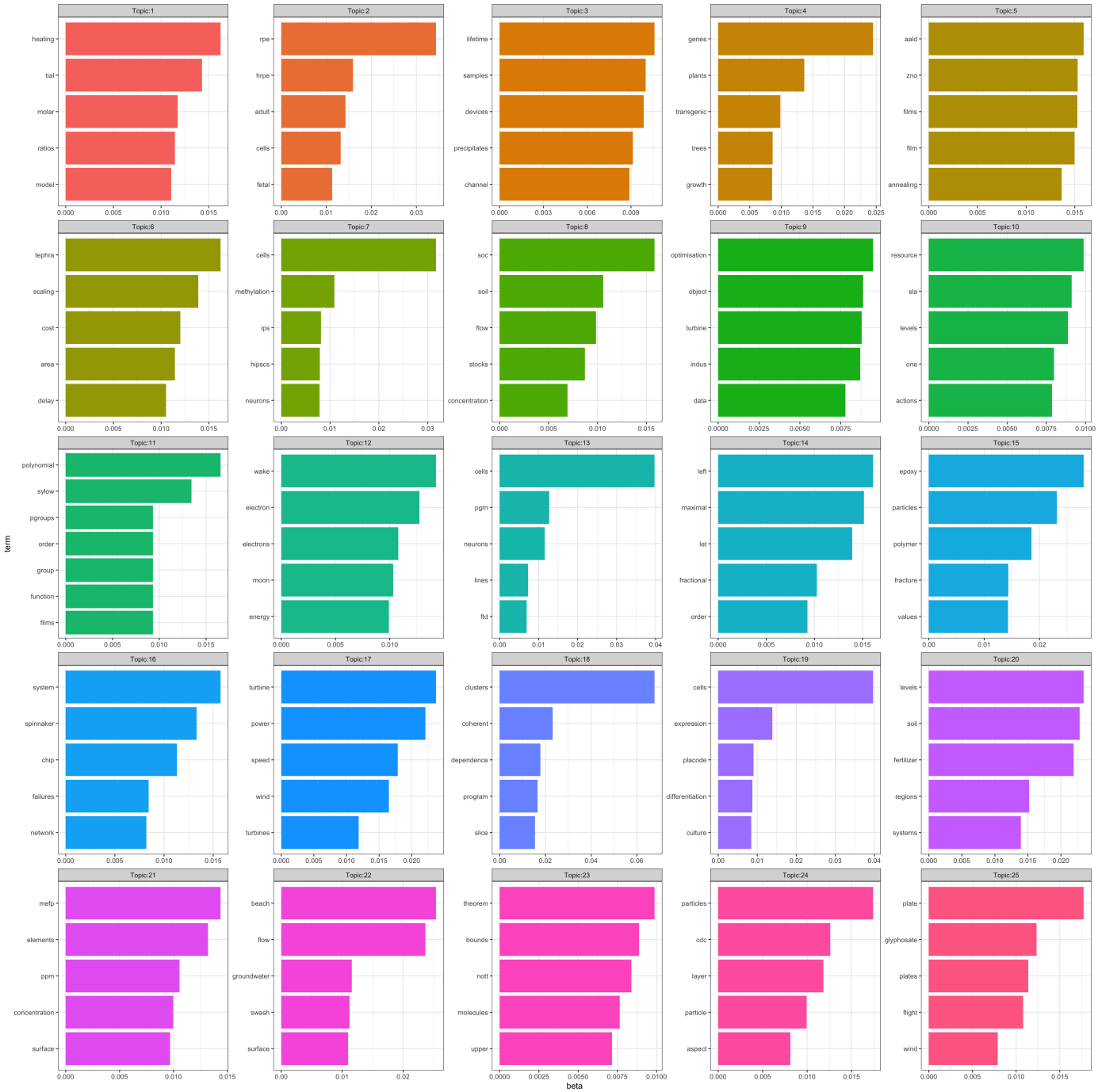
```
##     user   system  elapsed
##   73.576   12.708 1137.230
```

```
# Find Topic Number
FindTopicsNumber_plot(tunes)
```

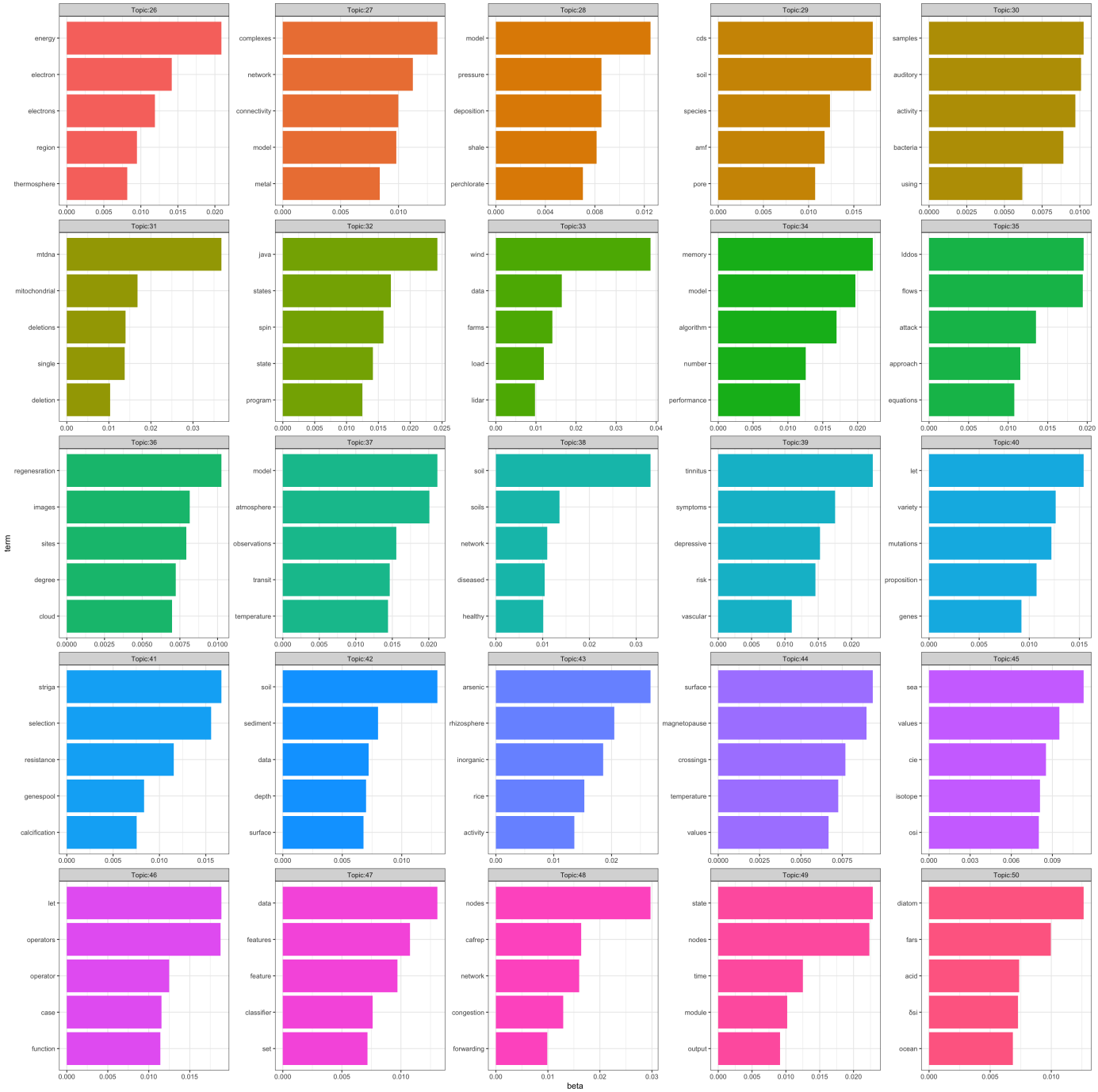



```
# Specifying the best LDA from the results above
best_LDA <- LDA(dtm, k = 50, control = list(seed = 332))
```

```
# Re-specifying the function for best usees
topic_analyze_final <- function(mod, topic_num) {


  # Creating a dataframe
  mod_df <- mod %>%
    tidy(matrix = "beta") %>%
    filter(topic %in% topic_num) %>%
    group_by(topic) %>%
    top_n(5, beta) %>%
    ungroup() %>%
    arrange(topic, -beta)

  # Creating a ggplot aesthetic
  plot <- mod_df %>%
    mutate(term = reorder_within(term, beta, topic)) %>%
    mutate(
    topic = paste0("Topic:", topic),
    num = parse_number(topic),
    topic = reorder(topic, num)
     ) %>%
    ggplot(aes(x = term, y = beta, fill = factor(topic))) +
    geom_col(show.legend = FALSE) +
    scale_x_reordered() +
    facet_wrap(~topic, scales = "free") +
    coord_flip()

  # return
  return(plot)
  }


# Showing the result
topic_analyze_final(best_LDA, 1:25)
```




```
topic_analyze_final(best_LDA, 26:50)
```

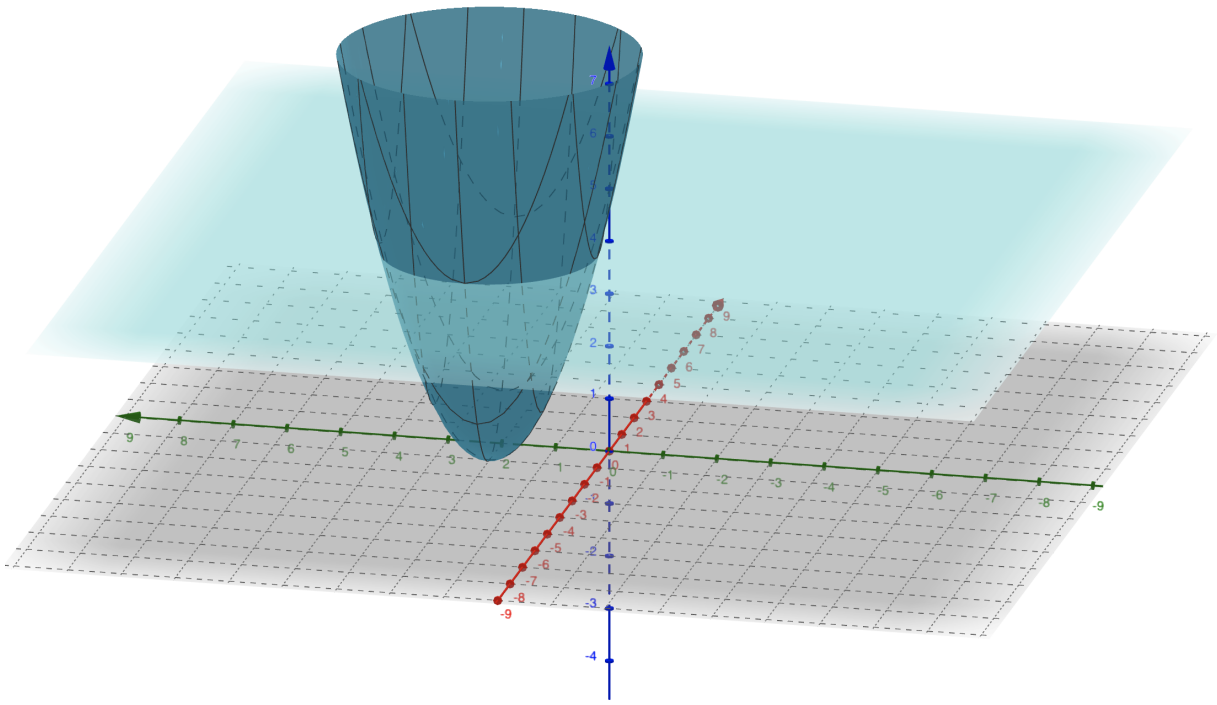


Question 2

a. Sketch the curve

$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$

```
knitr::include_graphics("/Users/boseongyun/Desktop/ml/PSET 4/pic1.png")
```



b. On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$

as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4$$

- **Answer: The set of points that satisfy $(1 + X_1)^2 + (2 - X_2)^2 > 4$ are the points that are above the plane. The set of points that satisfy $(1 + X_1)^2 + (2 - X_2)^2 \leq 4$ are the points that are below the plane (including the set of points that meet the plane)**

c. Suppose that a classifier assigns an observation to the blue class if

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$

and the red class otherwise. To what class is the observation (0, 0) classified? (-1,1)? (2,2)? (3,8)?

- **(0, 0) will be classified as blue**
- **(-1, 1) will be classified as red**
- **(2, 2) will be classified as blue**
- **(3, 8) will be classified as blue**

d. Argue that while the decision boundary in (c) is not linear in terms of X_1 and X_2, it is linear in terms of X_1, X_1^2, X_2, X_2^2

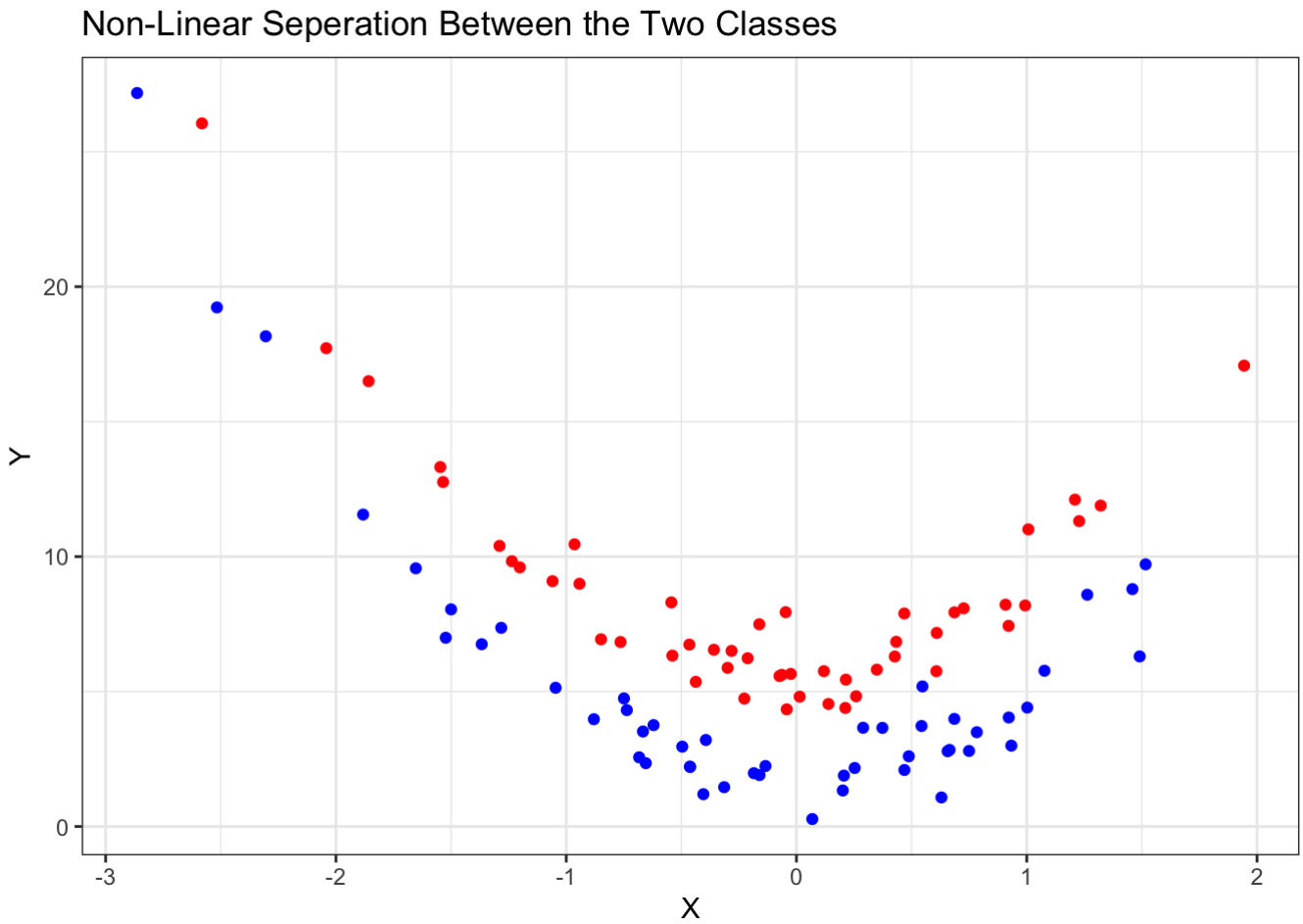
- Answer: **The decision boundary in (c) is not linear in terms of X_1 and X_2. It is linear in terms of X_1, X_1^2, X_2, X_2^2 because expanding $(1 + X_1)^2 + (2 - X_2)^2$ results in $X_1^2 + 2X_1 + X_2^2 - 4X_2 + 5$. That is, a linear combination of X_1, X_1^2, X_2, X_2^2 can determinethe decision boundary while a linear combination of X_1 and X_2 cannot. For instance, $X_1^2 + 2X_1 + X_2^2 - 4X_2 + 5$ where the coeffeicents represent the following form of $\beta_0 + \beta_1 X_1^2 + \beta_2 X_1 + \beta_3 X_2^2 + \beta_4 X_2$ is not a linear combination of β_1, β_2. Thus, the decision boundary in (c) is not linear in terms of X_1 and X_2, but linear in terms of X_1, X_1^2, X_2, X_2^2**

3. Generate a simulated two-class data set with 100 observations and two features in which there is a visible but non-linear separation between the two classes. Show that in this setting, a support vector

machine with a polynomial kernel (with degree greater than 1) or a radial kernel will outperform a linear support vector classifier (i.e., with a trivial kernel that just keeps the inner product) on the training data. Which technique performs best on the test data? Make plots and report training and test error rates in order to back up your assertions. (Hint: (1) Create a random initial dataset which lies along the parabola $y = 3 * x^2 + 4$, then separate the two classes by translating them along the y-axis. (2) Create both training and testing data frames by taking half of positive and negative classes and creating a new z vector of 0 and 1 for classes. (3) Train a model.)

- Answer: I have shown the performance of a support vector machine with different kernels on the training data. It does show that the radial kernel and polynomial kernel outperforms the linear support vector classifer on the training data. Accordingly, the raidal kernel method performs the best on the testing data as well. The radial kernel has the lowest 2% of the test error rate (a.k.a 1 misclassification)!

```
# setting the seed
set.seed(777)

# Creating the index for training
index <- sample(100, 50)

# 1. Create a random initial dataset which lies along the parabola
# the +2 or -2 are used to create the non-linear decision boundary!
x <- rnorm(100)
y <- 3 * x^2 + 4 + rnorm(100)
y[index] <- y[index] + 2 # +2 for distinction
y[-index] <- y[-index] - 2 # -2 for distinction

# Plotting
tibble(
  x1 = x[index],
  x0 = x[-index],
  y1 = y[index],
  y0 = y[-index]
) %>%
  ggplot() +
  geom_point(aes(x = x1, y = y1), color = "red") +
  geom_point(aes(x = x0, y = y0), color = "blue") +
  labs(
    x = "X",
    y = "Y",
    title = "Non-Linear Seperation Between the Two Classes"
  )
```

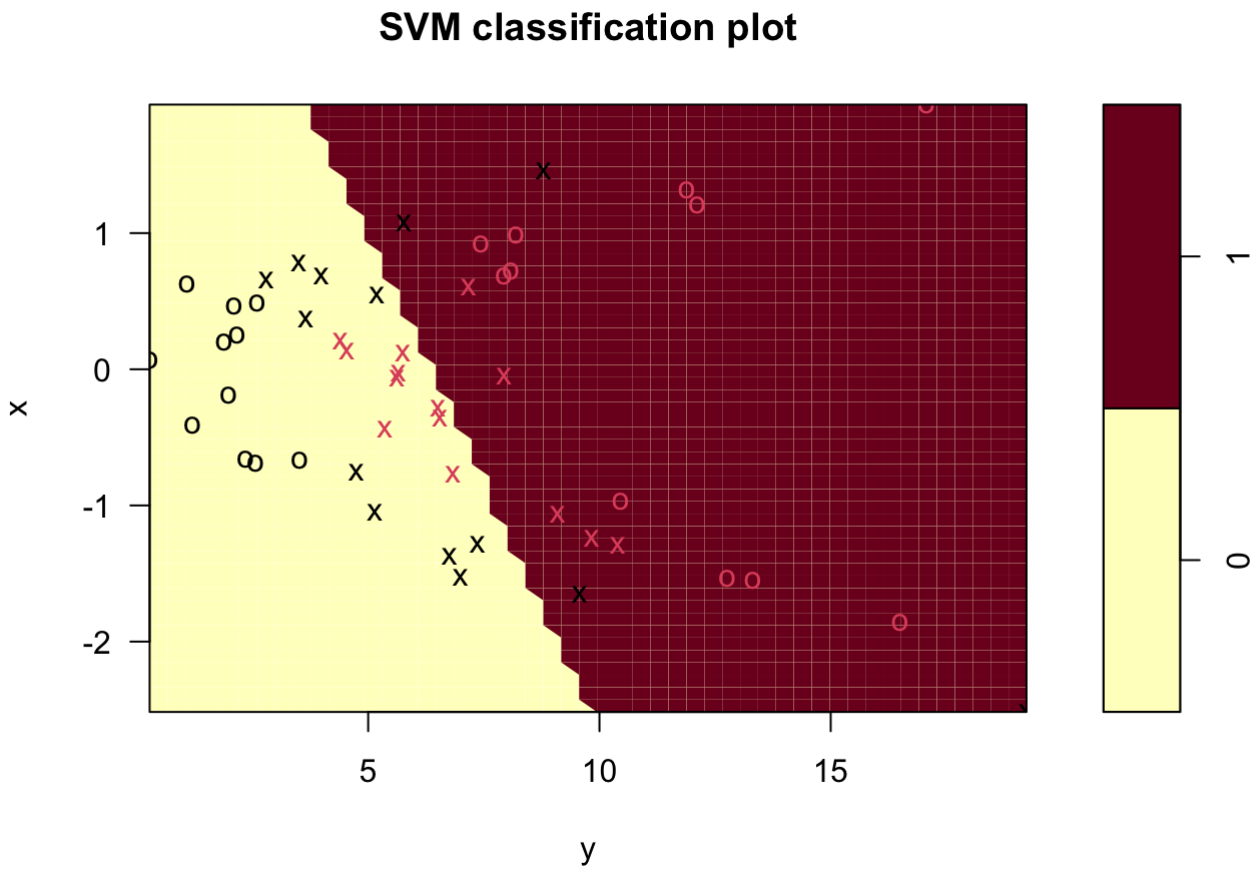



```
# 2. Createa both training and testing dataframes by taking half of positive and nega
tive classes
final_index <- c(sample(index, size = 25), sample(setdiff(1:100, index), size = 25))


# Creating a new z vector for classess
z <- rep(0, 100); z[index] <- 1

# creating the dataframes
df_train <- tibble(
  x = x[final_index],
  y = y[final_index],
  z = factor(z[final_index])
)

df_test <- tibble(
  x = x[-final_index],
  y = y[-final_index],
  z = factor(z[-final_index])
)
```

```
# 3. Train a model & show the results  ------------

# Creating a customized function for finding the training error
find_train_error <- function(mod, new_data) {

  # Creating the dataframe
  result_df <- predict(mod, new_data)  %>%
    as_tibble() %>%
    bind_cols(z[final_index]) %>%
    set_names("pred", "truth") %>%
    mutate(Correct = pred == truth) %>%
    count(Correct) %>%
    mutate(Rate = n / sum(n)) %>%
    set_names("Correct", "Count", "Proportion")

  # Returning the dataframe
  return(result_df)

}

find_test_error <- function(mod, new_data) {

  # Creating the dataframe
  result_df <- predict(mod, new_data)  %>%
    as_tibble() %>%
    bind_cols(z[-final_index]) %>%
    set_names("pred", "truth") %>%
    mutate(Correct = pred == truth) %>%
    count(Correct) %>%
    mutate(Rate = n / sum(n)) %>%
    set_names("Correct", "Count", "Proportion")

  # Returning the dataframe
  return(result_df)

}
```

- SVM (Linear Kernel with cost = 5 and other parameters set to default settings): The following plot and table shows that the classifier makes 13 errors with the error rate of 26% on the training data

```
# Training Cases

# Linear SVM
svm_linear <- svm(z ~ .,
                  data = df_train,
                  kernel = "linear",
                  cost = 5)
plot(svm_linear, df_train)
```

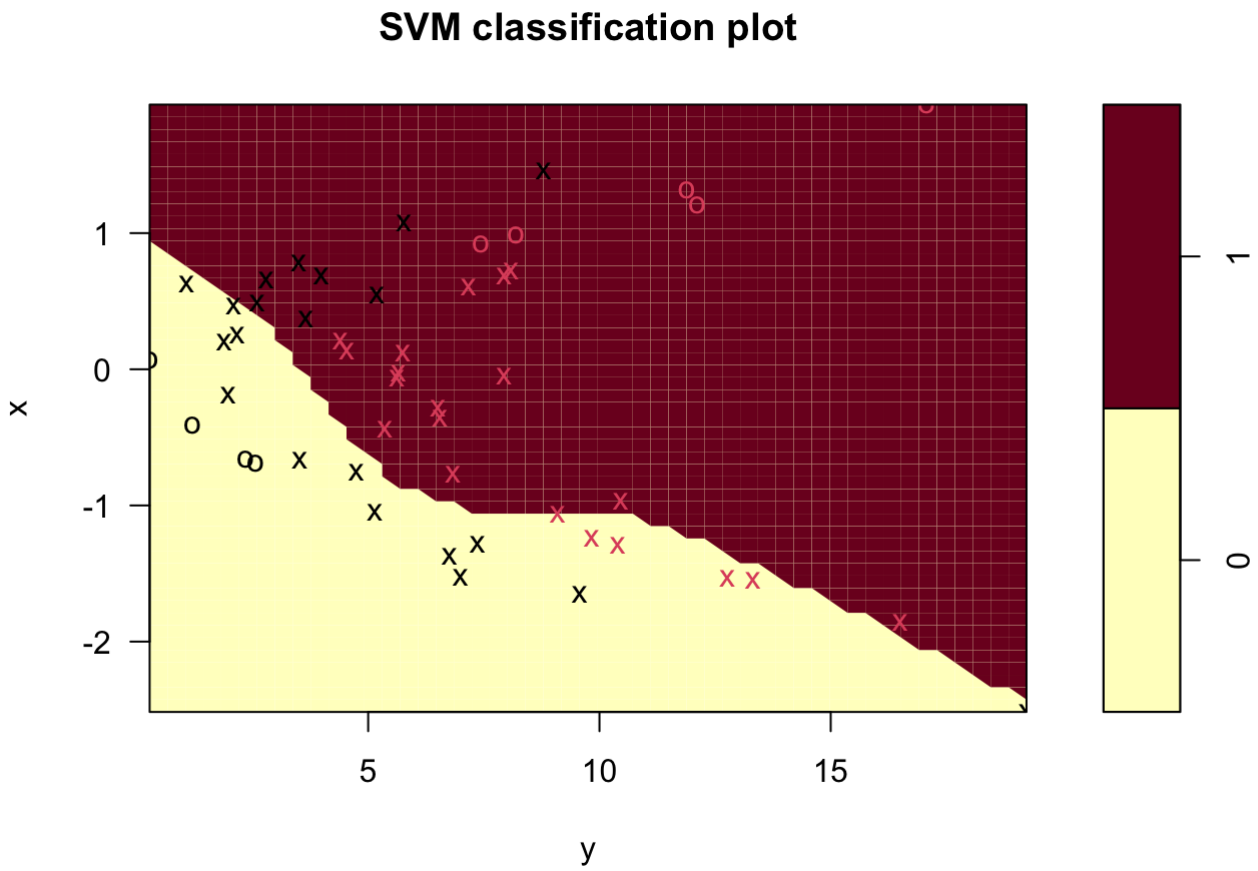



```
find_train_error(svm_linear, df_train)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE      13       0.26
## 2 TRUE       37       0.74
```

- SVM (Polynomial Kernel with cost = 5 and degree set to 3): The following plot and table shows that the classifier makes 12 errors with the error rate of 24% on the training data.

```
# Polynomial
svm_polynomial <- svm(z ~ .,
                      data = df_train,
                      kernel = "polynomial",
                      cost = 5)
plot(svm_polynomial, df_train)
```

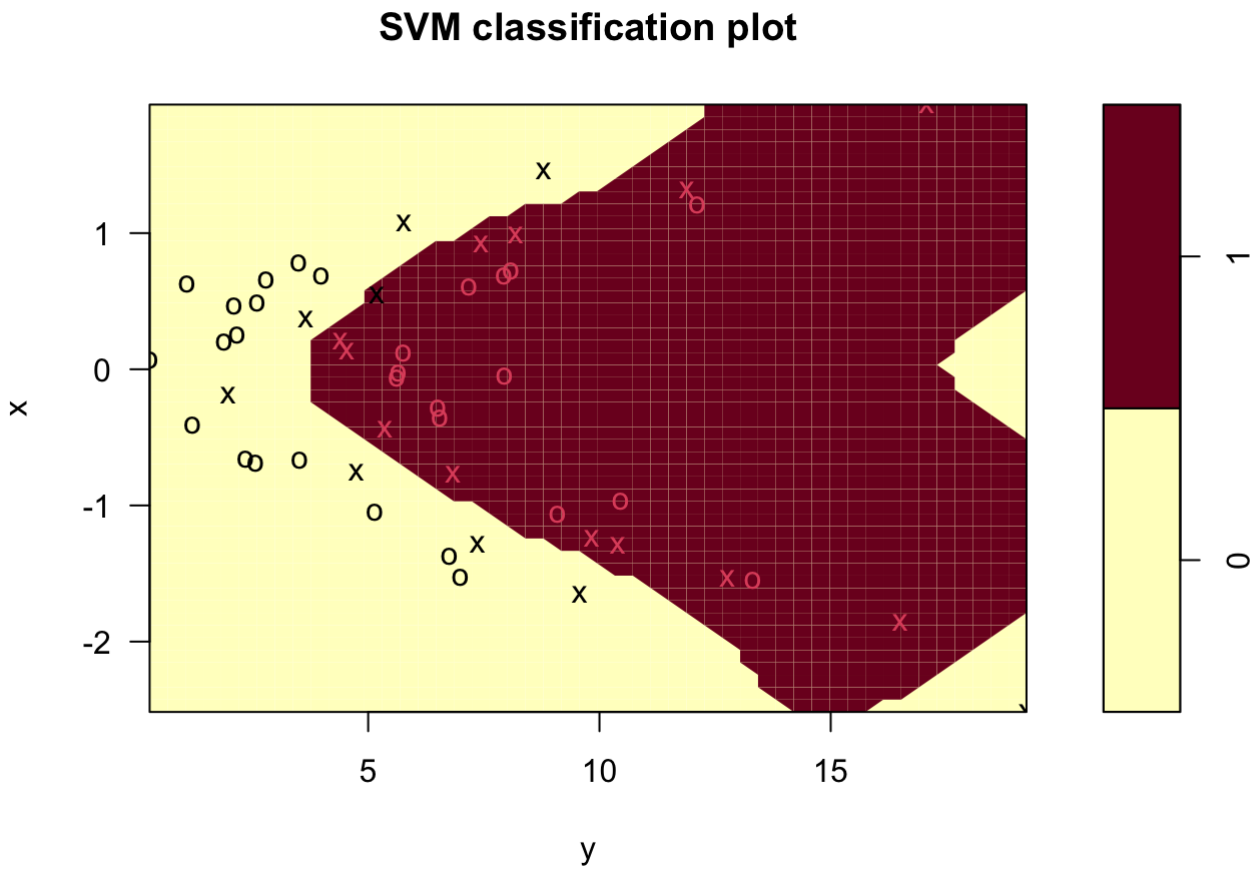



```
find_train_error(svm_polynomial, df_train)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE      12       0.24
## 2 TRUE       38       0.76
```

- SVM (Radial Kernel with cost = 5 and gamma = 1): The following plot and table shows that the classifier makes 1 errors with the error rate of 2% on the training data

```
# Radial
svm_radial <- svm(z ~ .,
                  data = df_train,
                  kernel = "radial",
                  gamma = 1,
                  cost = 5)
plot(svm_radial, df_train)
```

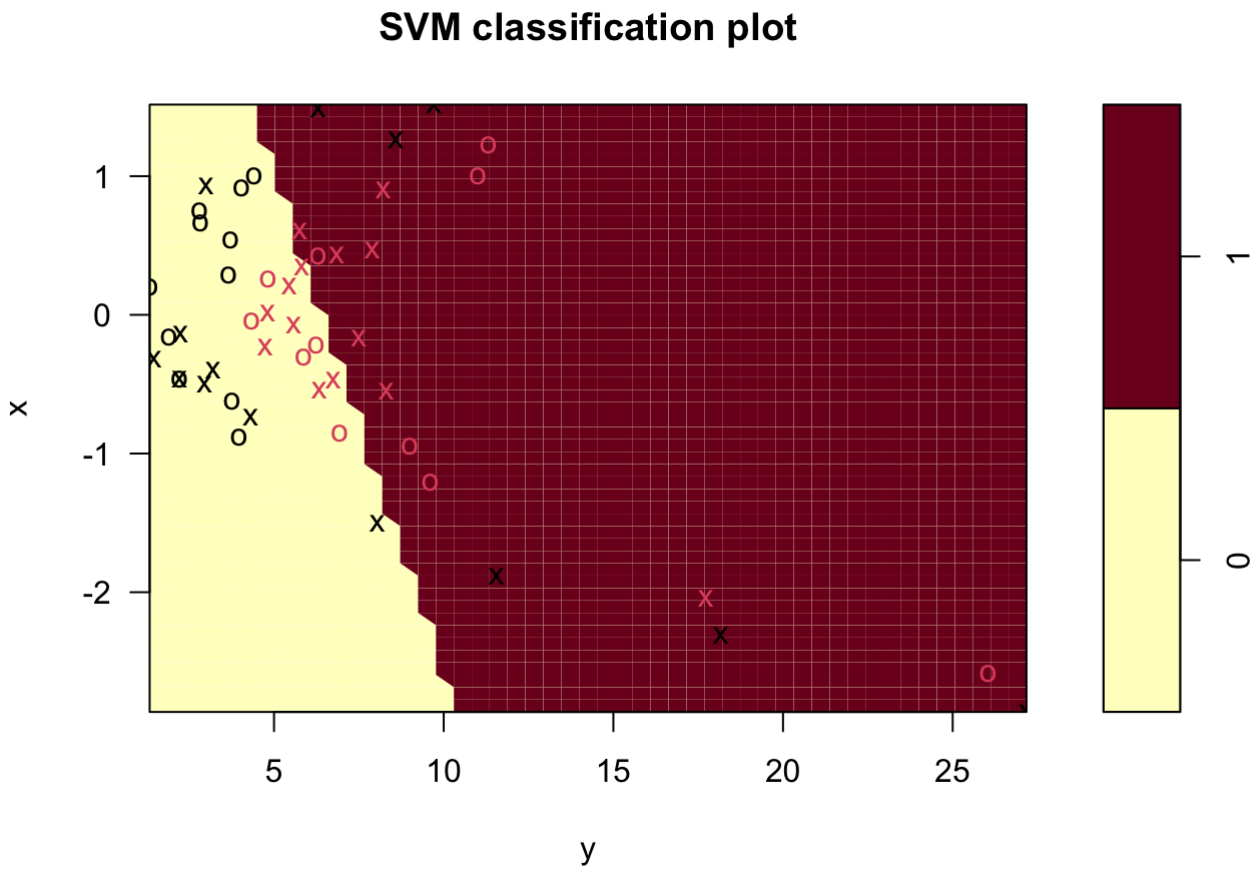



```
find_train_error(svm_radial, df_train)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE       1       0.02
## 2 TRUE       49       0.98
```

- SVM (Linear Kernel with cost = 5): The following plot and table shows that the classifier makes 17 errors with the error rate of 34% on the testing data

```
# Test Cases
plot(svm_linear, df_test)
```

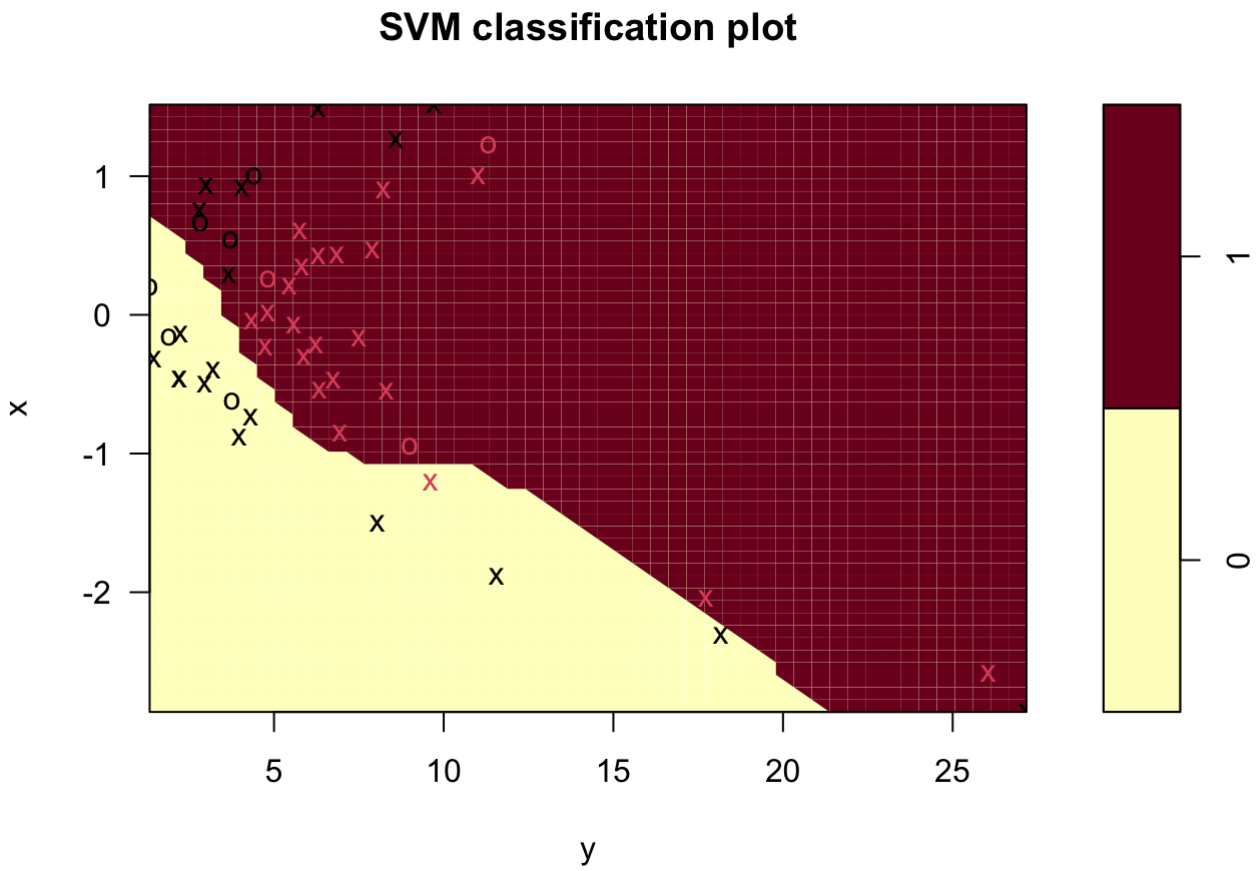



```
find_test_error(svm_linear, df_test)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE      17       0.34
## 2 TRUE       33       0.66
```

- SVM (Polynomial Kernel with cost = 5 and degree set to 3): The following plot and table shows that the classifier makes 12 errors with the error rate of 24% on the testing data

```
# Polynomial
plot(svm_polynomial, df_test)
```

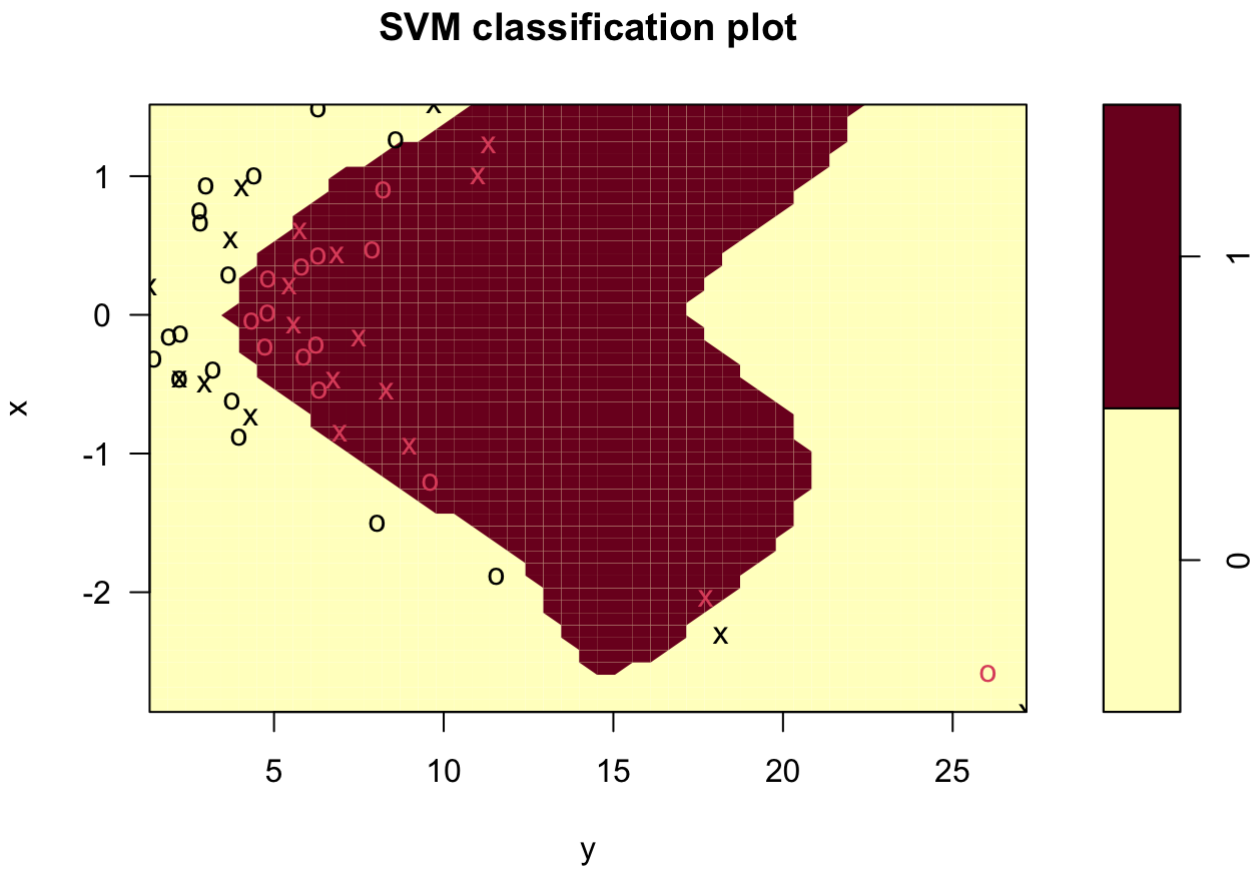



```
find_test_error(svm_polynomial, df_test)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE      12       0.24
## 2 TRUE       38       0.76
```

- SVM (Radial Kernel with cost = 5 and gamma = 1): The following plot and table shows that the classifier makes 1 error with the error rate of 2% on the testing data

```
# Radial
plot(svm_radial, df_test)
```




```
find_test_error(svm_radial, df_test)
```

```
## # A tibble: 2 x 3
##   Correct Count Proportion
## * <lgl>   <int>      <dbl>
## 1 FALSE       1       0.02
## 2 TRUE       49       0.98
```

4. In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the Auto data set.

a. Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```
# Auto dataset
auto <- ISLR::Auto %>%
  mutate(mpg_class = factor(ifelse(mpg > median(mpg), 1, 0)))
```

b. Fit a support vector classifier (with just the inner product as the kernel) to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

- The cross-validated errors are the lowest when the cost is set to 1 for the SVM with linear basis kernels.

```
# Tuning the costs
tune_res_linear <- tune(
  svm,
  mpg_class ~.,
  data = auto,
  ranges = tibble(cost = c(0.01, 0.05, 0.1, 0.5, 1, 10, 30, 50, 100)),
  kernel = "linear"
)

summary(tune_res_linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01538462
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07416667 0.04106796
## 2 5e-02 0.05884615 0.04207264
## 3 1e-01 0.04346154 0.02981791
## 4 5e-01 0.01794872 0.03209399
## 5 1e+00 0.01538462 0.02477158
## 6 1e+01 0.01788462 0.01727588
## 7 3e+01 0.03064103 0.02359501
## 8 5e+01 0.03064103 0.02359501
## 9 1e+02 0.03064103 0.02359501
```

```
# Best SVM linear
best_linear <- svm(mpg_class ~.,
                   data = auto,
                   kernel = "linear",
                   cost = 1)
```

c. Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost (from the e1071 R package). Comment on your results.

- The cross-validated errors are the lowest when the cost is set to 100 and gamma is set to 0.01 for the SVM with radial basis kernels.
- The cross-validated errors are the lowest when the cost is set to 100 and degree is set to 1 for the SVM with polynomial basis kernels.

```
# Radial kernels
tune_res_radial <- tune(
  svm,
  mpg_class ~.,
  data = auto,
  kernel = "radial",
  ranges = list(
    cost = c(0.01, 0.05, 0.1, 0.5, 1, 10, 30, 50, 100),
    gamma = c(0.01, 0.05, 0.1, 0.5, 1, 10, 50, 100)
  )
)

summary(tune_res_radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100  0.01
##
## - best performance: 0.01282051
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-02 1e-02 0.56365385 0.05401551
## 2  5e-02 1e-02 0.08948718 0.05433169
## 3  1e-01 1e-02 0.08948718 0.05433169
## 4  5e-01 1e-02 0.07666667 0.04507200
## 5  1e+00 1e-02 0.07153846 0.04304927
## 6  1e+01 1e-02 0.02294872 0.02807826
## 7  3e+01 1e-02 0.01538462 0.01792836
## 8  5e+01 1e-02 0.01794872 0.02110955
## 9  1e+02 1e-02 0.01282051 0.01813094
## 10 1e-02 5e-02 0.17128205 0.10848213
## 11 5e-02 5e-02 0.08173077 0.04771349
## 12 1e-01 5e-02 0.07666667 0.04507200
## 13 5e-01 5e-02 0.06897436 0.04510440
## 14 1e+00 5e-02 0.05621795 0.04141999
## 15 1e+01 5e-02 0.02044872 0.02020886
## 16 3e+01 5e-02 0.02301282 0.01891104
## 17 5e+01 5e-02 0.02557692 0.02093679
## 18 1e+02 5e-02 0.02051282 0.02356248
## 19 1e-02 1e-01 0.22198718 0.10338615
## 20 5e-02 1e-01 0.08173077 0.04771349
## 21 1e-01 1e-01 0.07666667 0.04507200
## 22 5e-01 1e-01 0.06647436 0.04040219
## 23 1e+00 1e-01 0.05365385 0.04253806
## 24 1e+01 1e-01 0.03064103 0.02637448
## 25 3e+01 1e-01 0.03839744 0.03024912
## 26 5e+01 1e-01 0.03839744 0.03024912
## 27 1e+02 1e-01 0.03583333 0.03245677
## 28 1e-02 5e-01 0.56365385 0.05401551
## 29 5e-02 5e-01 0.26717949 0.15535255
## 30 1e-01 5e-01 0.08179487 0.04793112
## 31 5e-01 5e-01 0.06647436 0.03855171
## 32 1e+00 5e-01 0.04089744 0.04388392
## 33 1e+01 5e-01 0.04339744 0.04350592
## 34 3e+01 5e-01 0.04339744 0.04350592
## 35 5e+01 5e-01 0.04339744 0.04350592
## 36 1e+02 5e-01 0.04339744 0.04350592
## 37 1e-02 1e+00 0.56365385 0.05401551
## 38 5e-02 1e+00 0.56365385 0.05401551
## 39 1e-01 1e+00 0.56365385 0.05401551
## 40 5e-01 1e+00 0.07160256 0.04648116
## 41 1e+00 1e+00 0.05621795 0.04641038
## 42 1e+01 1e+00 0.05878205 0.04834923
## 43 3e+01 1e+00 0.05878205 0.04834923
## 44 5e+01 1e+00 0.05878205 0.04834923
```

```
## 45 1e+02 1e+00 0.05878205 0.04834923
## 46 1e-02 1e+01 0.56365385 0.05401551
## 47 5e-02 1e+01 0.56365385 0.05401551
## 48 1e-01 1e+01 0.56365385 0.05401551
## 49 5e-01 1e+01 0.56365385 0.05401551
## 50 1e+00 1e+01 0.53032051 0.05770659
## 51 1e+01 1e+01 0.52519231 0.06158300
## 52 3e+01 1e+01 0.52519231 0.06158300
## 53 5e+01 1e+01 0.52519231 0.06158300
## 54 1e+02 1e+01 0.52519231 0.06158300
## 55 1e-02 5e+01 0.56365385 0.05401551
## 56 5e-02 5e+01 0.56365385 0.05401551
## 57 1e-01 5e+01 0.56365385 0.05401551
## 58 5e-01 5e+01 0.56365385 0.05401551
## 59 1e+00 5e+01 0.56108974 0.05721076
## 60 1e+01 5e+01 0.55852564 0.05888763
## 61 3e+01 5e+01 0.55852564 0.05888763
## 62 5e+01 5e+01 0.55852564 0.05888763
## 63 1e+02 5e+01 0.55852564 0.05888763
## 64 1e-02 1e+02 0.56365385 0.05401551
## 65 5e-02 1e+02 0.56365385 0.05401551
## 66 1e-01 1e+02 0.56365385 0.05401551
## 67 5e-01 1e+02 0.56365385 0.05401551
## 68 1e+00 1e+02 0.56365385 0.05401551
## 69 1e+01 1e+02 0.56365385 0.05401551
## 70 3e+01 1e+02 0.56365385 0.05401551
## 71 5e+01 1e+02 0.56365385 0.05401551
## 72 1e+02 1e+02 0.56365385 0.05401551
```

```
# Best SVM Radial
best_radial <- svm(mpg_class ~.,
                   data = auto,
                   kernel = "radial",
                   ranges = list(cost = 100,
                                 gamma = 0.01)
                   )

# Polynomial basis kernels
tune_res_polynomial <- tune(
  svm,
  mpg_class ~.,
  data = auto,
  kernel = "polynomial",
  ranges = list(cost = c(0.01, 0.05, 0.1, 0.5, 1, 10, 30, 50, 100),
                degree = c(1:5))
  )

summary(tune_res_polynomial)
```

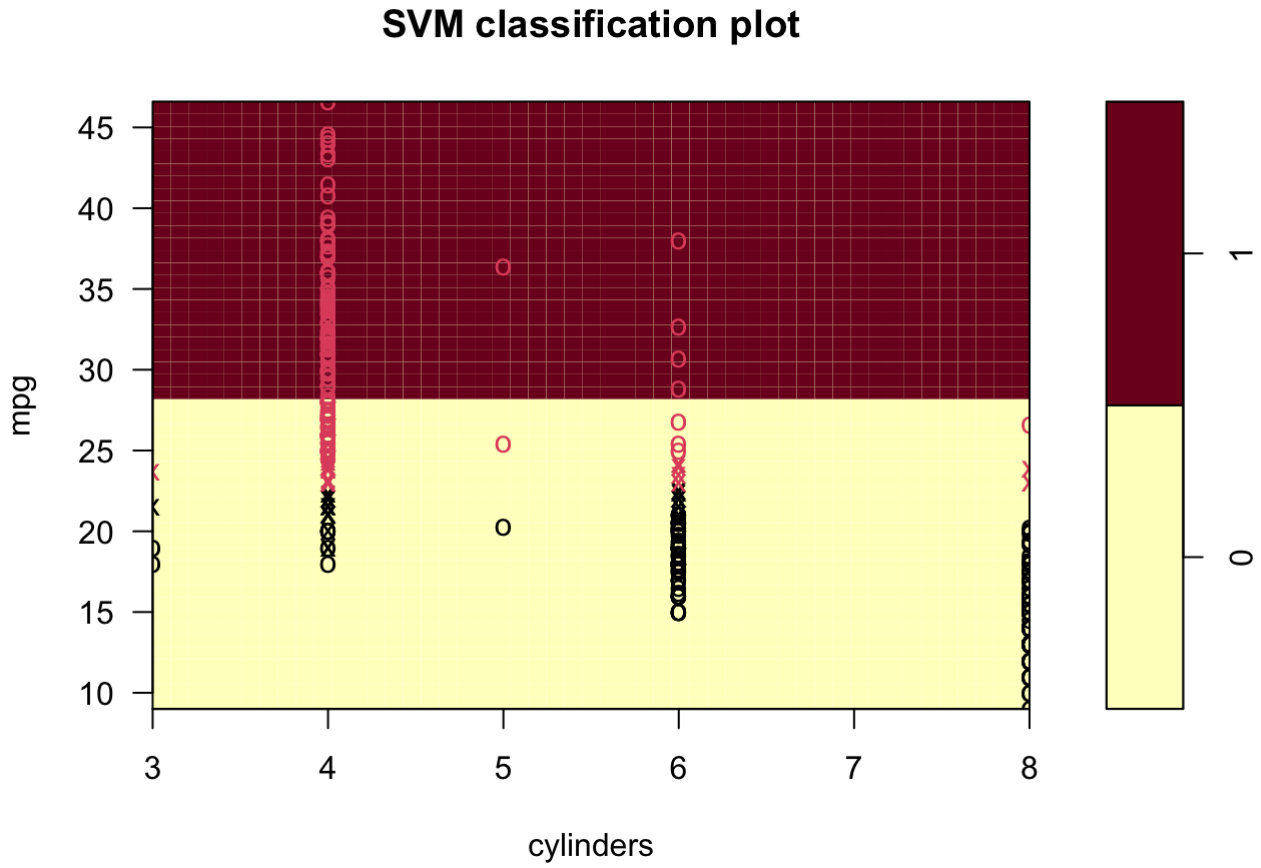
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   100      1
##
## - best performance: 0.02294872
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1  1e-02      1 0.56391026 0.04882098
## 2  5e-02      1 0.56391026 0.04882098
## 3  1e-01      1 0.17660256 0.10803908
## 4  5e-01      1 0.09192308 0.05142004
## 5  1e+00      1 0.07903846 0.04566860
## 6  1e+01      1 0.06634615 0.04555368
## 7  3e+01      1 0.04589744 0.03571919
## 8  5e+01      1 0.03826923 0.03654290
## 9  1e+02      1 0.02294872 0.02534336
## 10 1e-02      2 0.56391026 0.04882098
## 11 5e-02      2 0.56391026 0.04882098
## 12 1e-01      2 0.56391026 0.04882098
## 13 5e-01      2 0.56391026 0.04882098
## 14 1e+00      2 0.56391026 0.04882098
## 15 1e+01      2 0.53326923 0.07640761
## 16 3e+01      2 0.41602564 0.08889557
## 17 5e+01      2 0.34705128 0.09855186
## 18 1e+02      2 0.30102564 0.06481316
## 19 1e-02      3 0.56391026 0.04882098
## 20 5e-02      3 0.56391026 0.04882098
## 21 1e-01      3 0.56391026 0.04882098
## 22 5e-01      3 0.56391026 0.04882098
## 23 1e+00      3 0.56391026 0.04882098
## 24 1e+01      3 0.56391026 0.04882098
## 25 3e+01      3 0.54365385 0.06607676
## 26 5e+01      3 0.43173077 0.10132138
## 27 1e+02      3 0.34448718 0.10001242
## 28 1e-02      4 0.56391026 0.04882098
## 29 5e-02      4 0.56391026 0.04882098
## 30 1e-01      4 0.56391026 0.04882098
## 31 5e-01      4 0.56391026 0.04882098
## 32 1e+00      4 0.56391026 0.04882098
## 33 1e+01      4 0.56391026 0.04882098
## 34 3e+01      4 0.56391026 0.04882098
## 35 5e+01      4 0.56391026 0.04882098
## 36 1e+02      4 0.56391026 0.04882098
## 37 1e-02      5 0.56391026 0.04882098
## 38 5e-02      5 0.56391026 0.04882098
## 39 1e-01      5 0.56391026 0.04882098
## 40 5e-01      5 0.56391026 0.04882098
## 41 1e+00      5 0.56391026 0.04882098
## 42 1e+01      5 0.56391026 0.04882098
## 43 3e+01      5 0.56391026 0.04882098
```

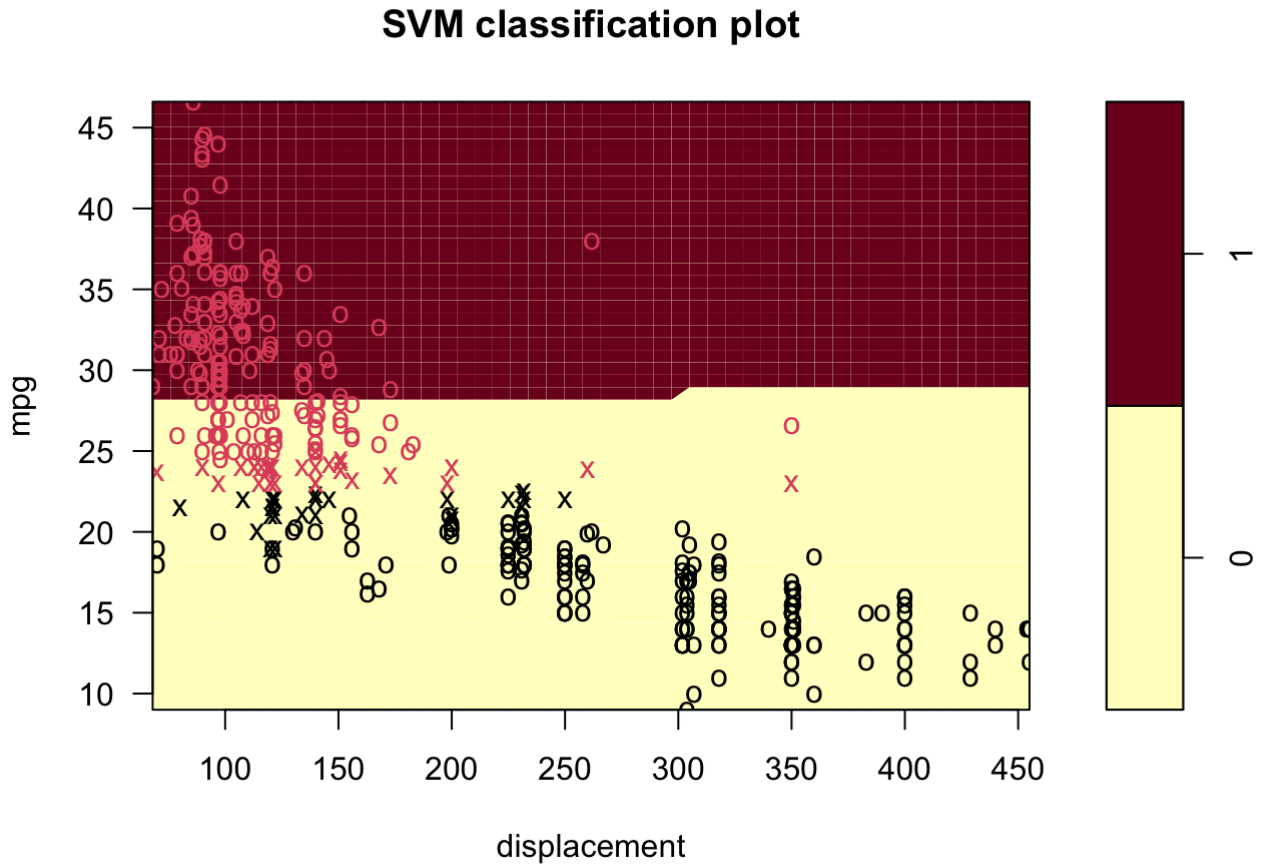
```
## 44 5e+01      5 0.56391026 0.04882098
## 45 1e+02      5 0.56391026 0.04882098
```

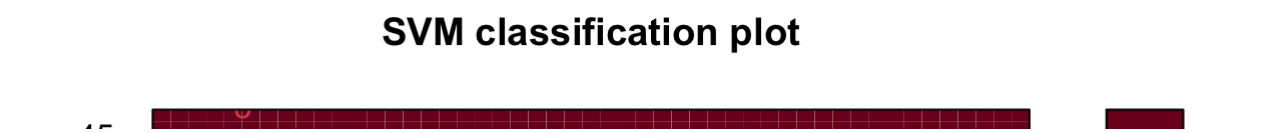
```
# Best SVM Polynomial
best_polynomial <- svm(mpg_class ~.,
                       data = auto,
                       kernel = "polynomial",
                       ranges = list(cost = 100,
                                     degree = 1)
                       )
```

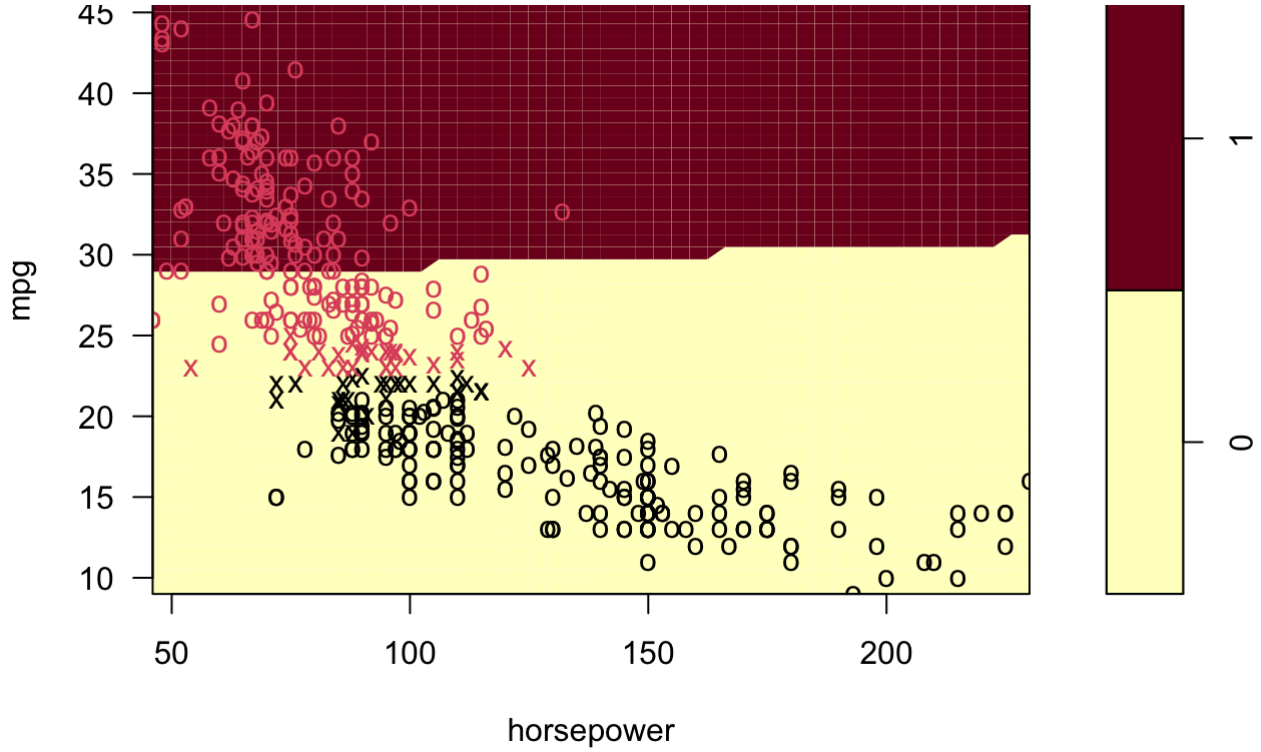
d. Make some plots to back up your assertions in (b) and (c). Hint: Use the plot function to create plots displaying pairs of variables at a time. For example, where svmfit contains your fitted model, and dat is a data frame containing your data, you can type: plot(svmfit, dat, x1~x4) in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type ?plot.svm.

```
plot_pair <- function(mod) {

  for(i in 1:7) {

    # Selecting meaningful variables
    selected_vars <- names(auto)[!(names(auto) %in% c("mpg", "name", "mpg_class"))]

    # Saving the name of the variable
    var <- selected_vars[i]

    # Specifying the formula
    formula <- as.formula(paste("mpg ~", var, sep = ""))

    # Saving the plot
    plot(mod, auto, formula)
  }
}


# Plotting!

# SVM: linear kernel
plot_pair(best_linear)
```

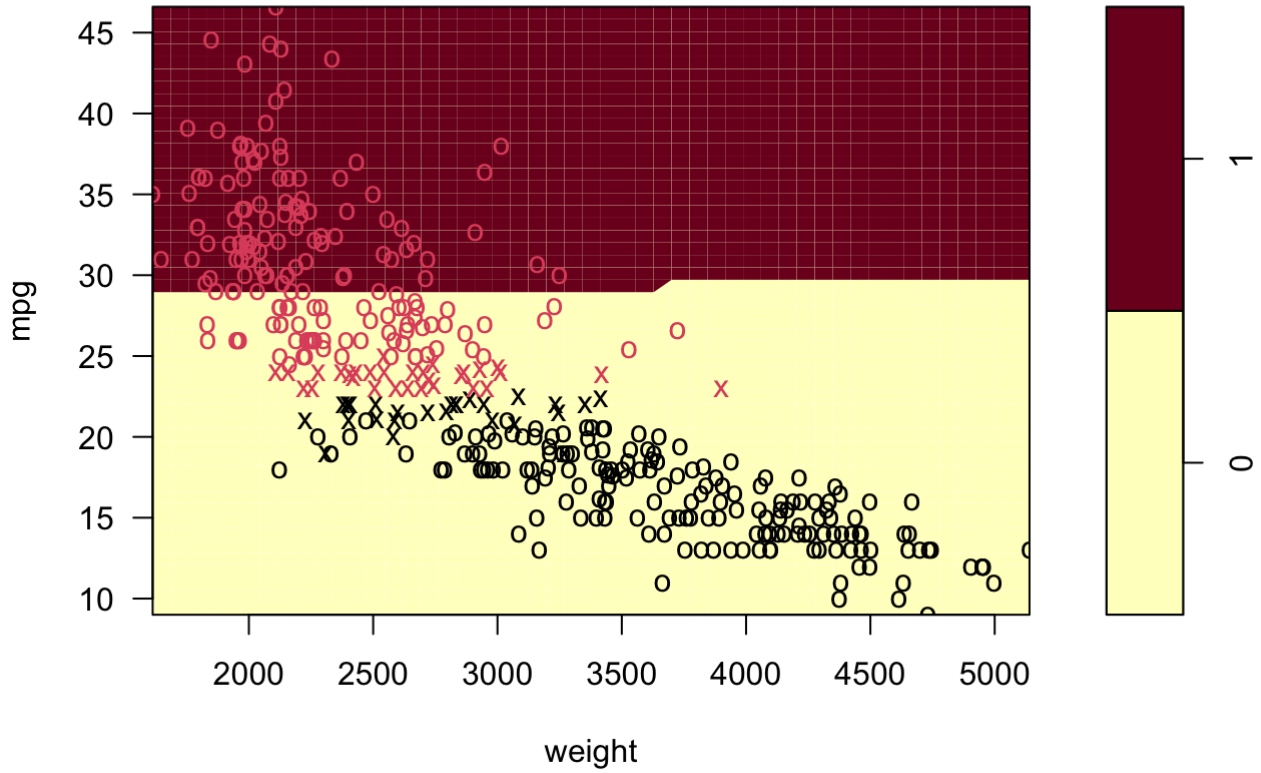

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

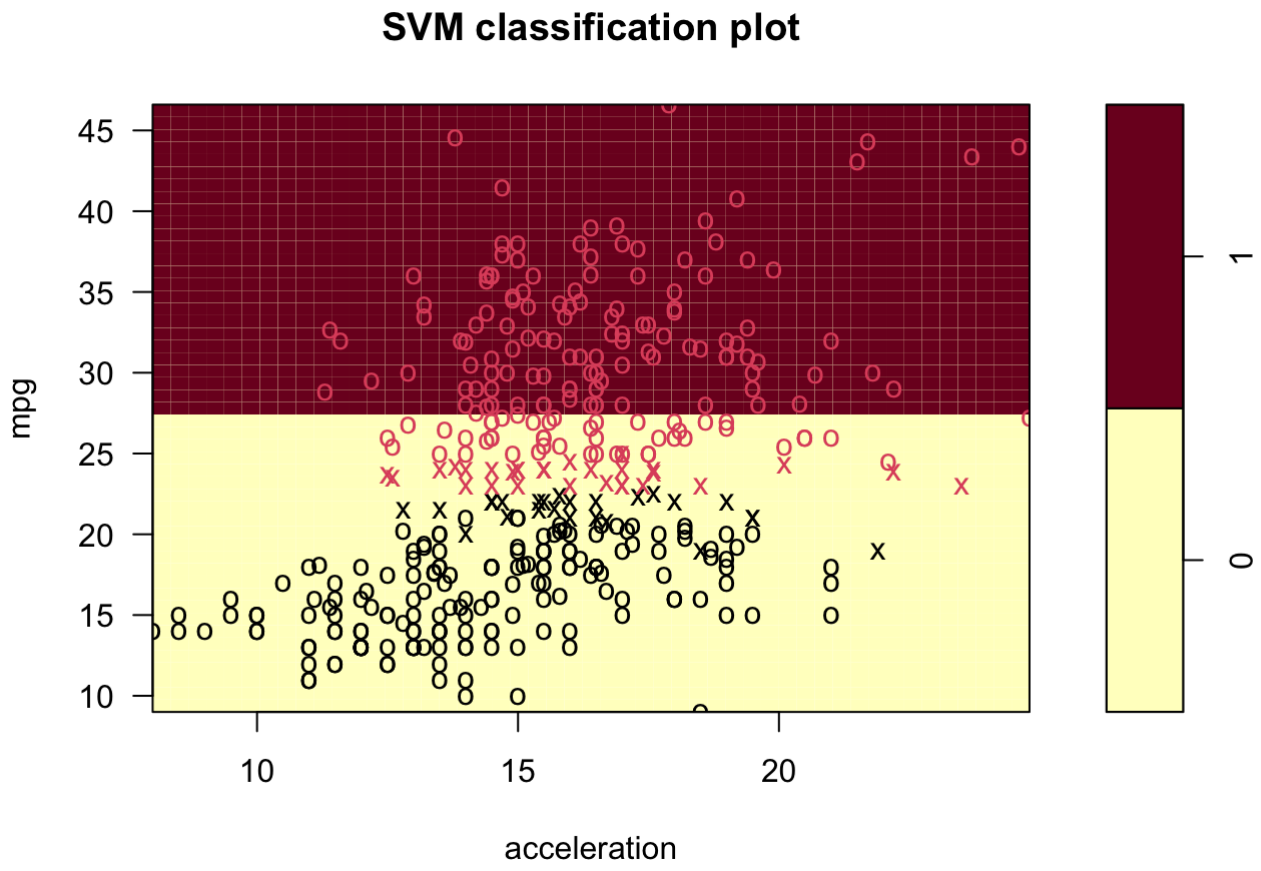

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0

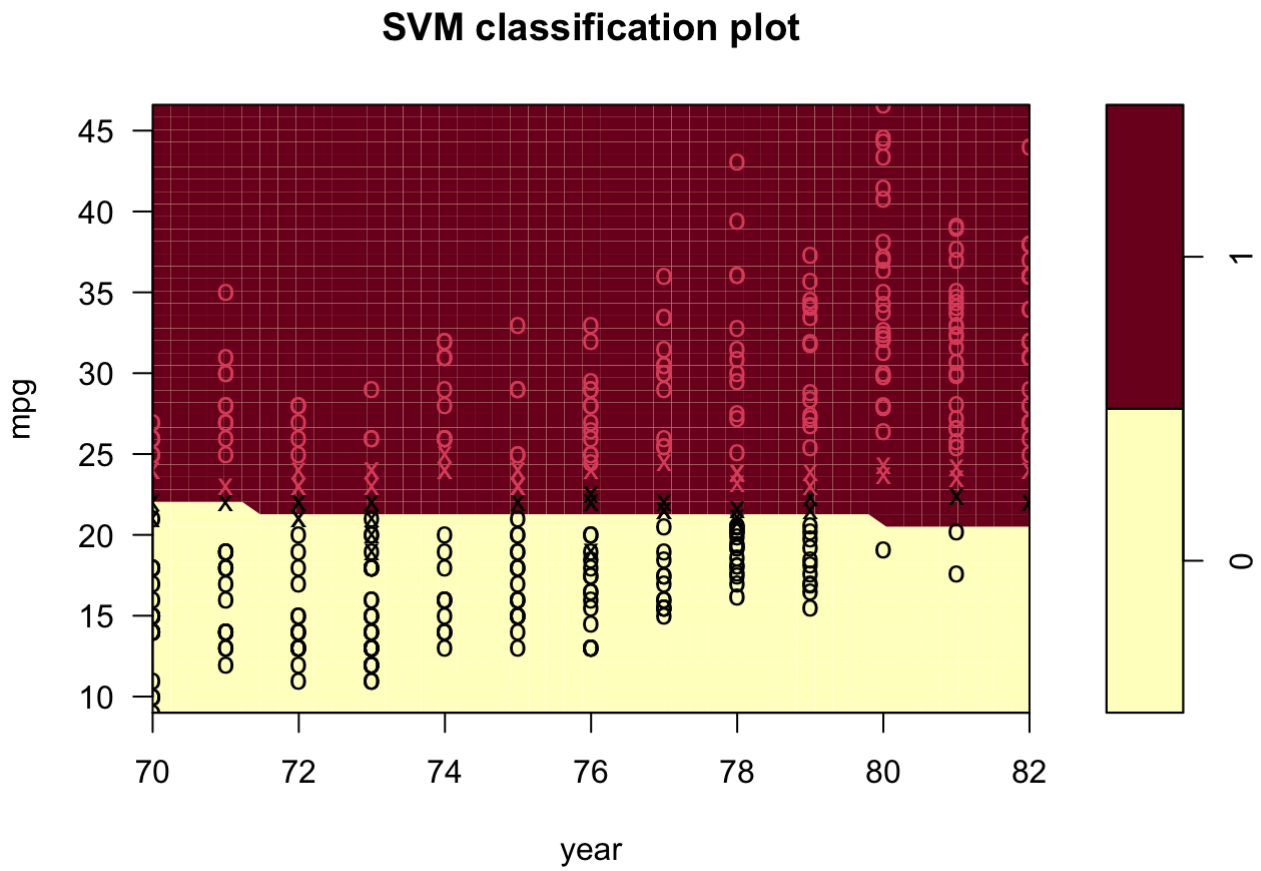

SVM classification plot

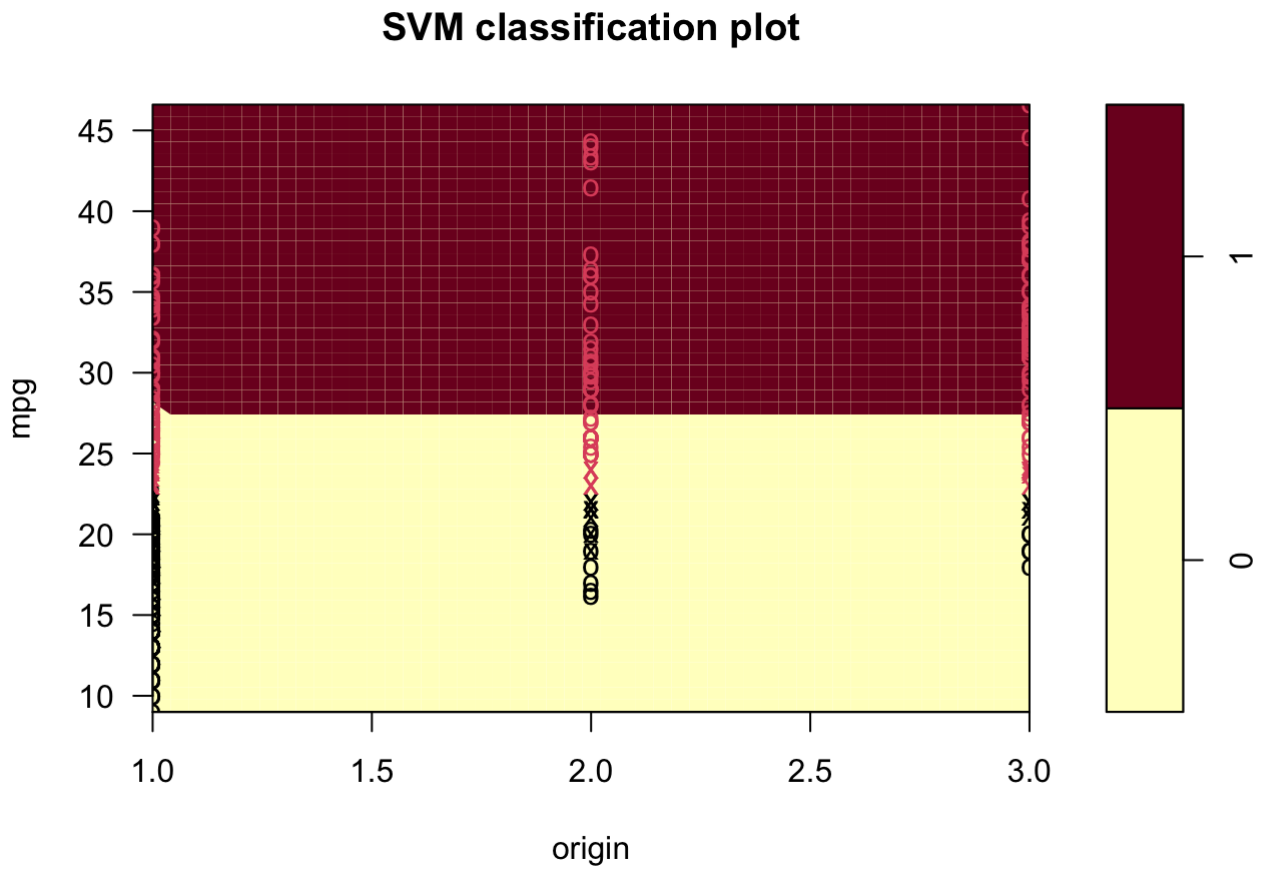


SVM classification plot

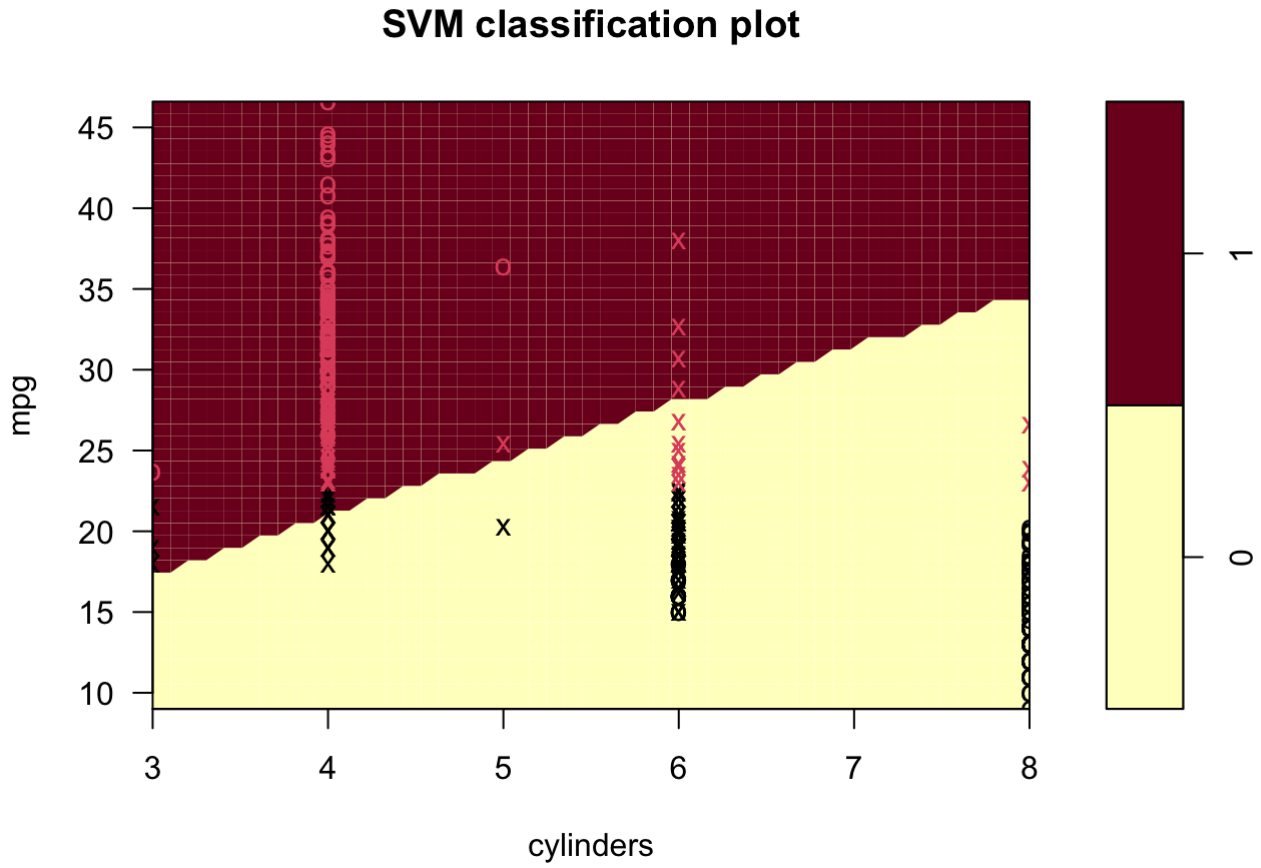


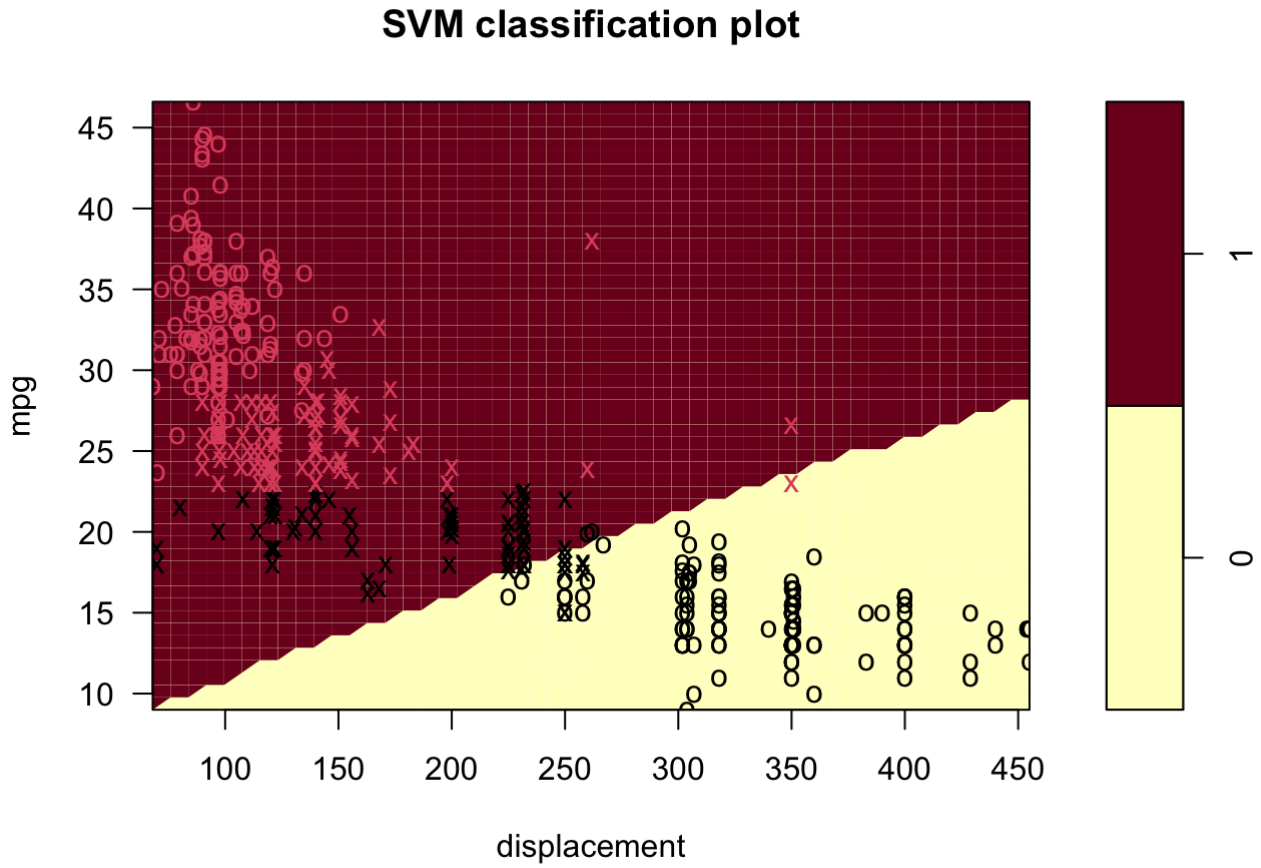

SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

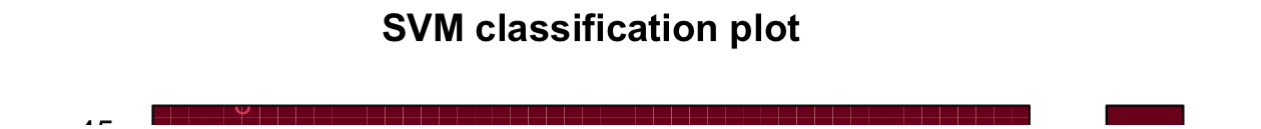

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0

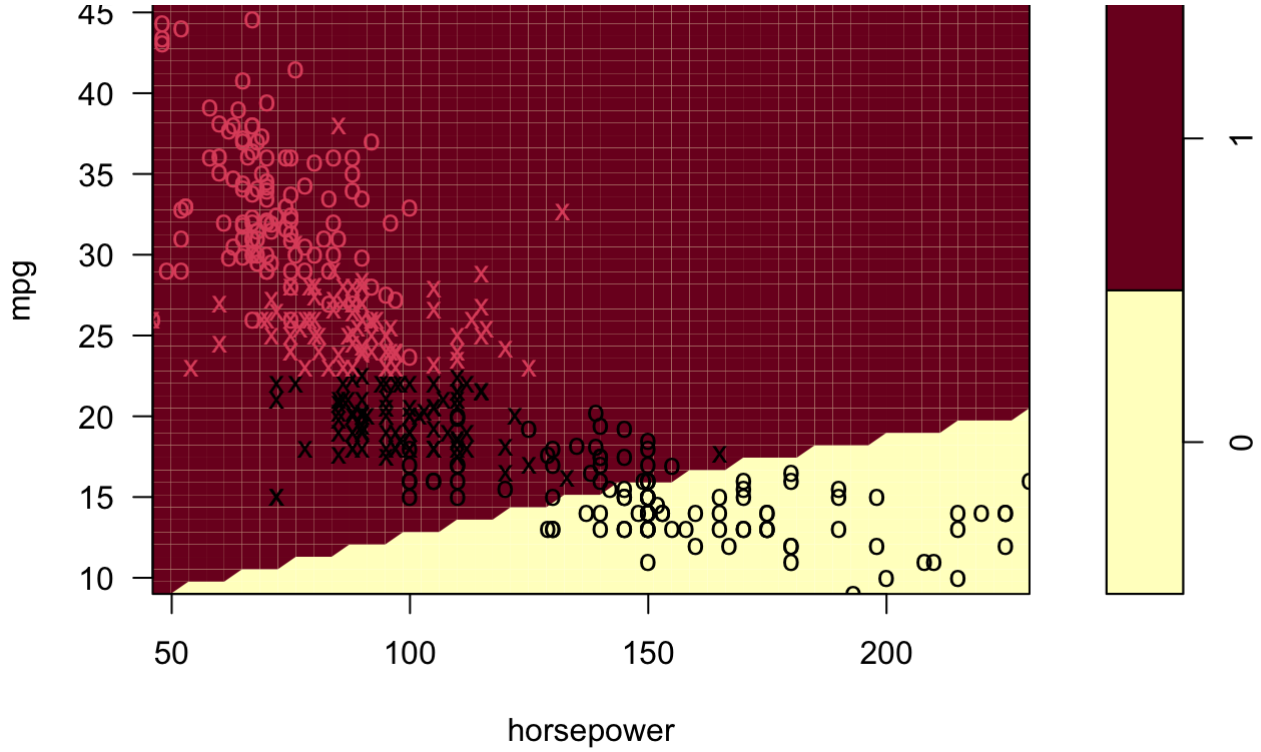



```
# SVM: radial kernel
plot_pair(best_radial)
```

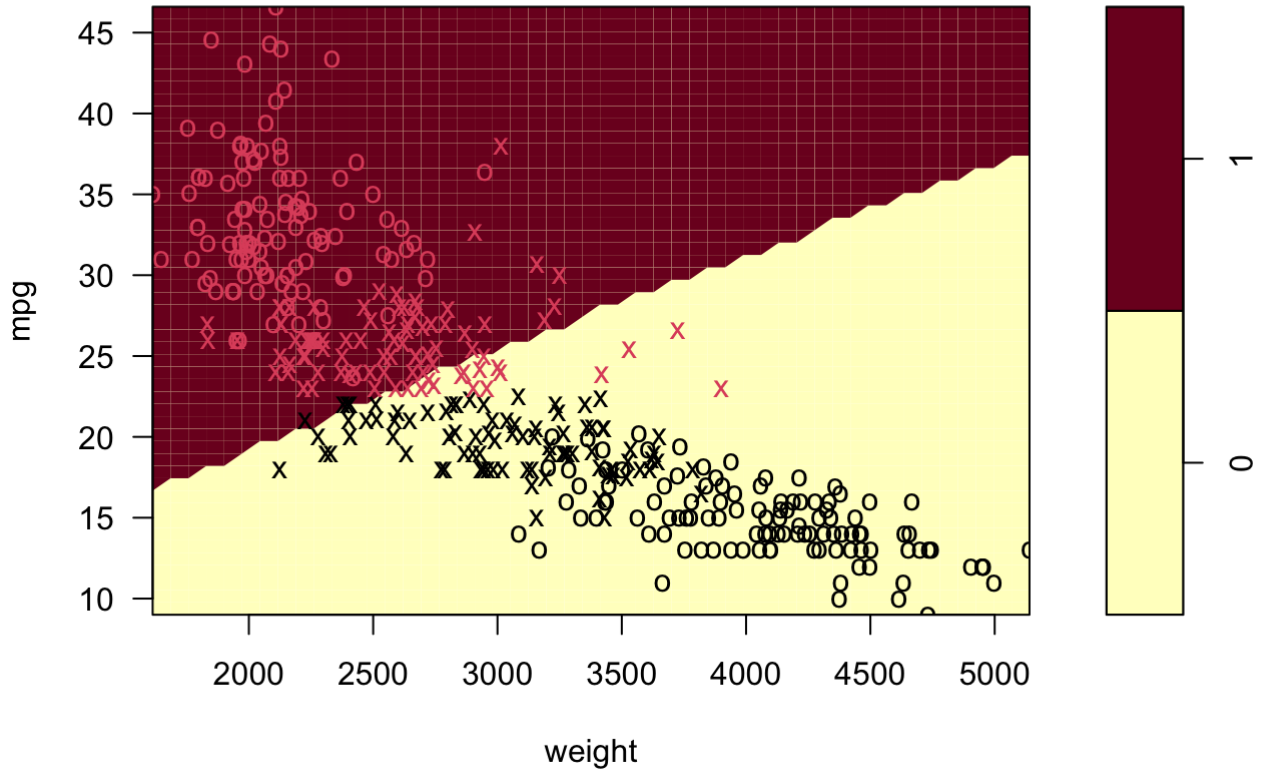

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

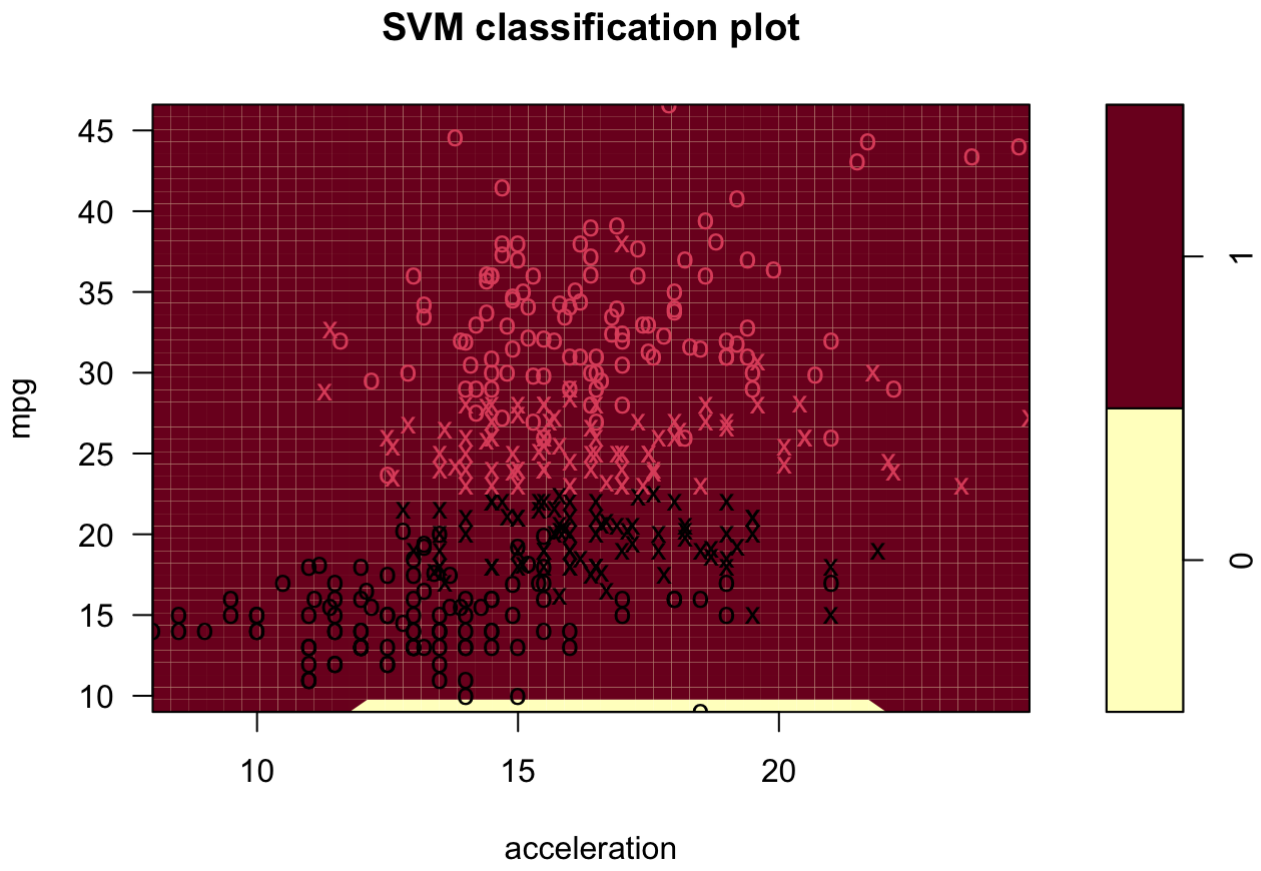

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0

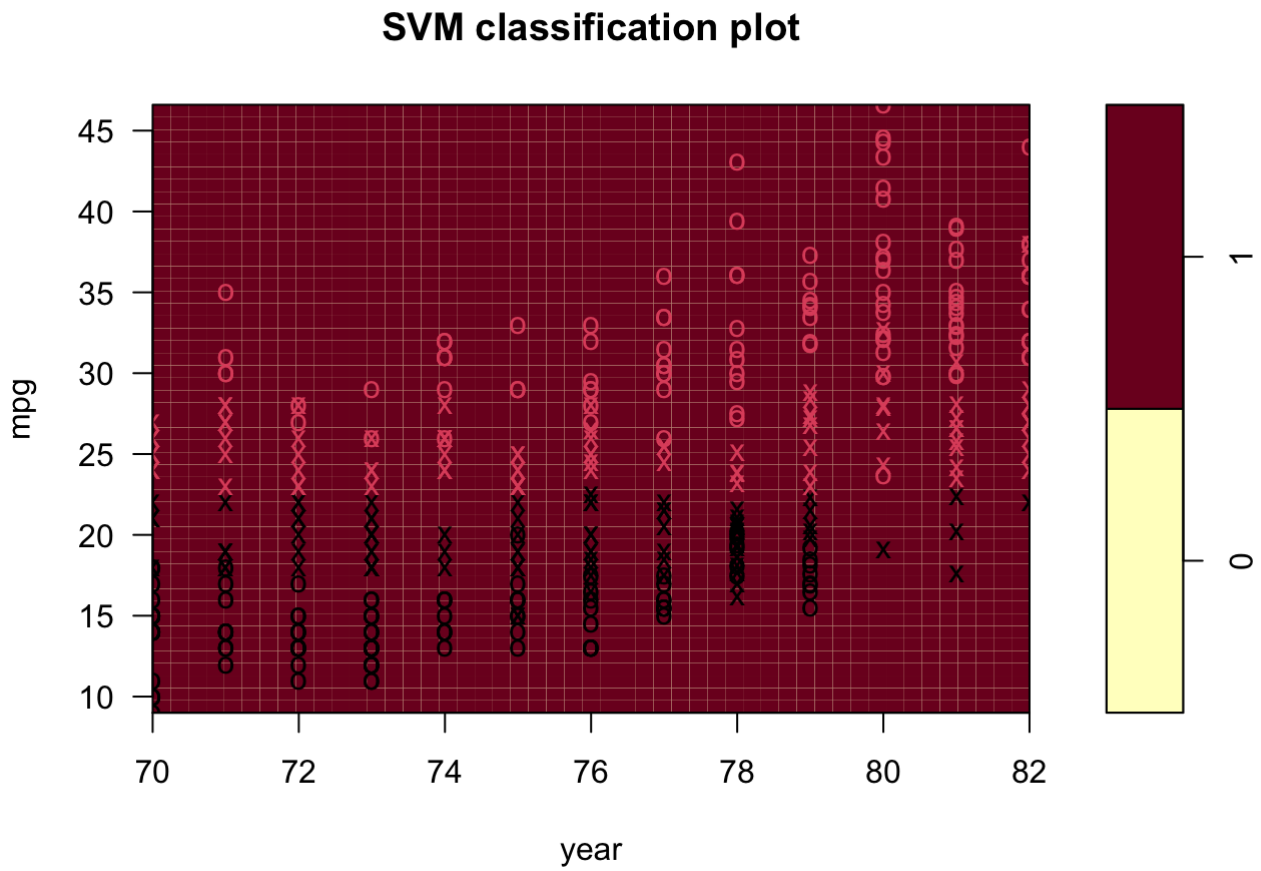

SVM classification plot

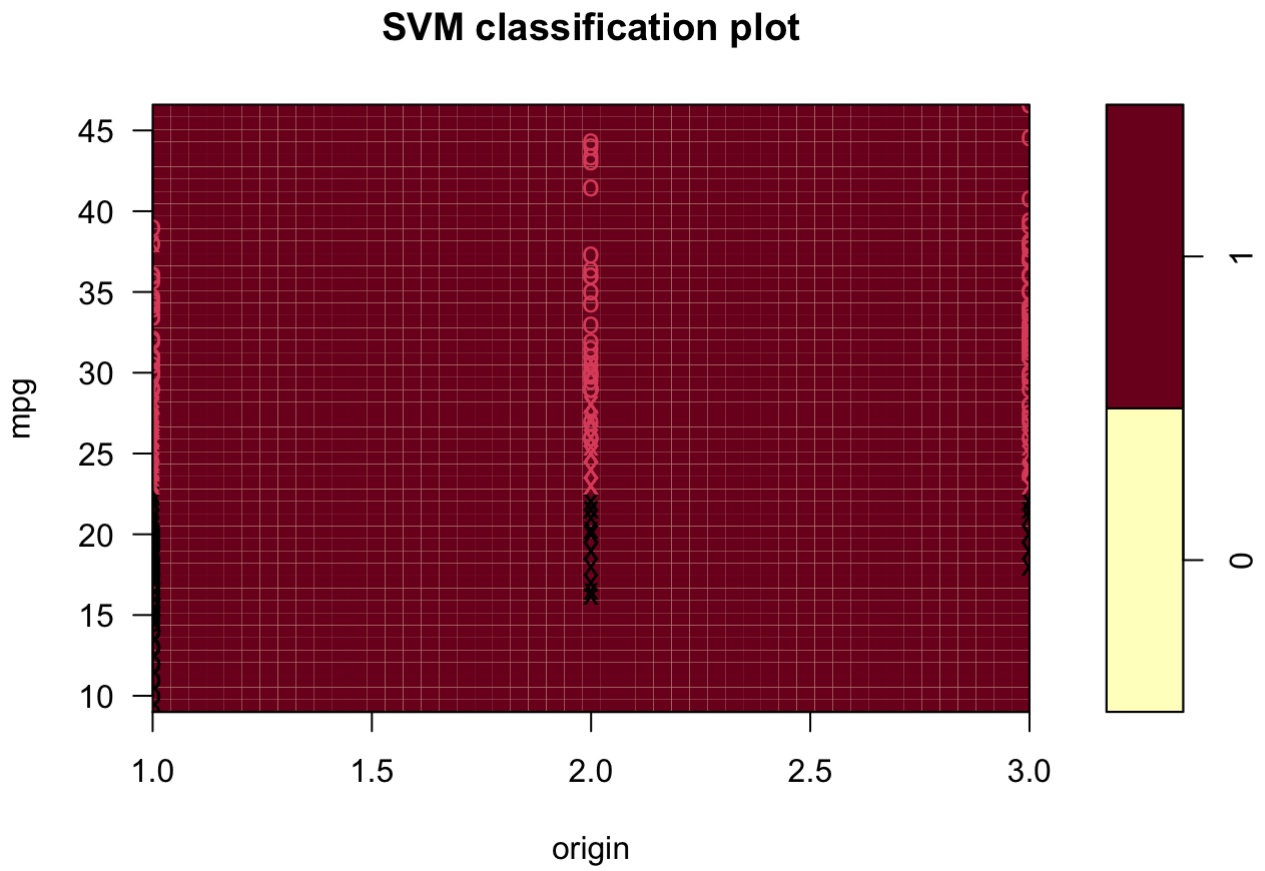


SVM classification plot

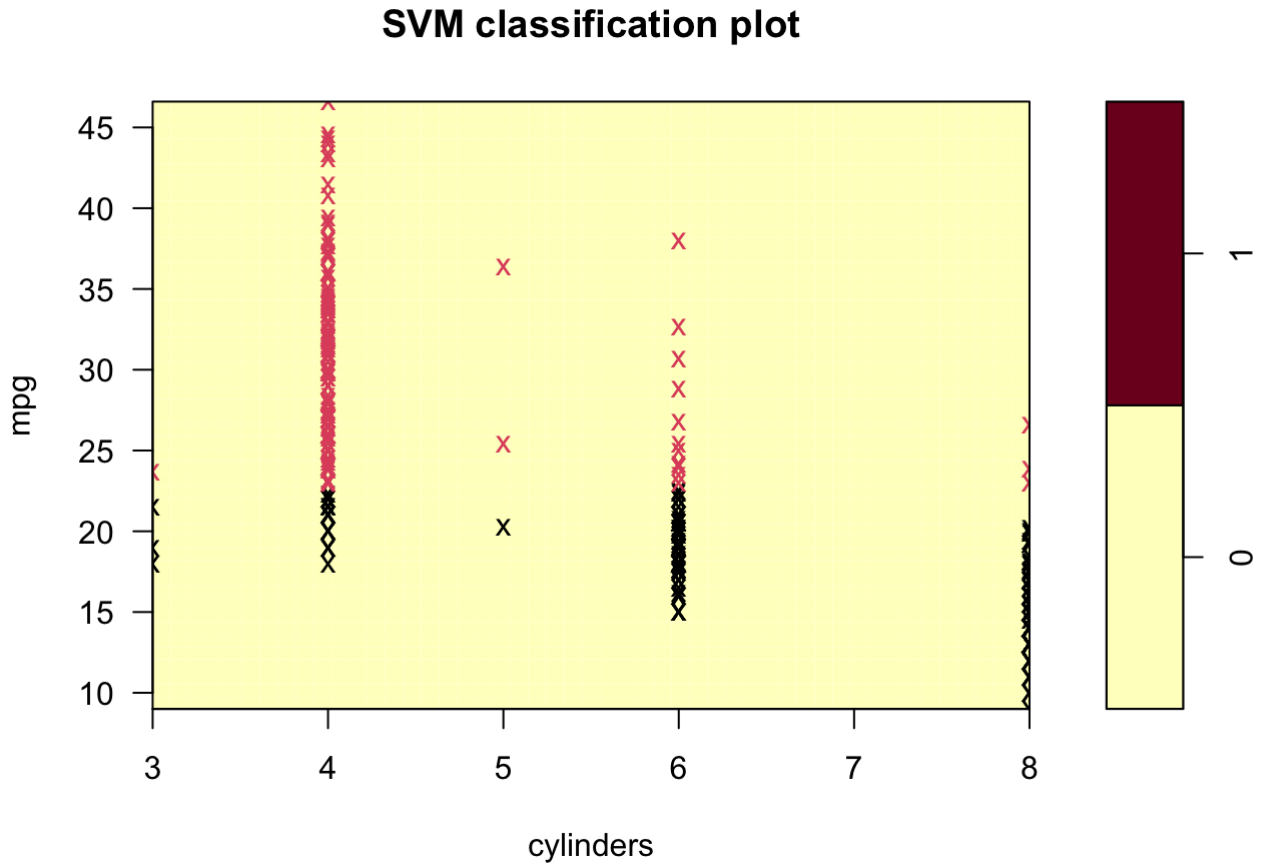


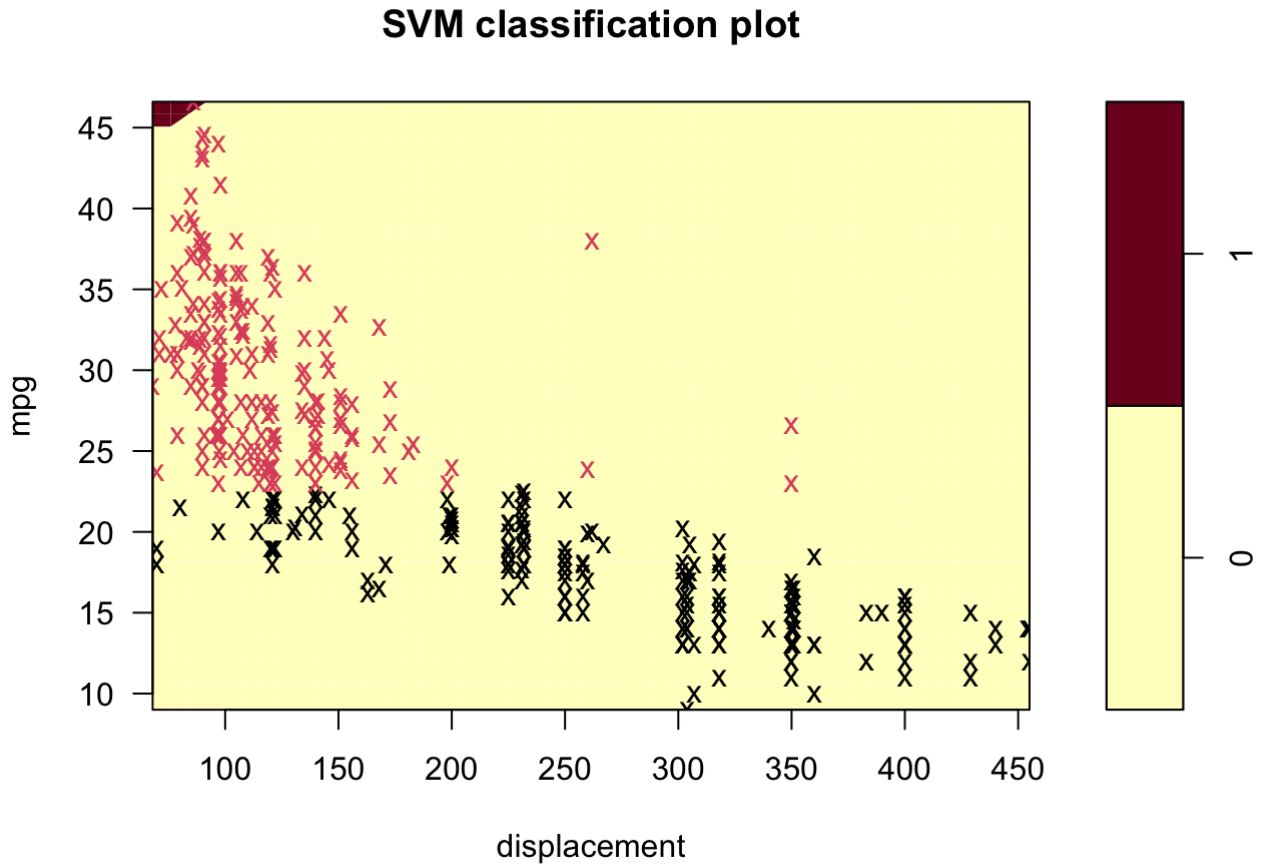

SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0

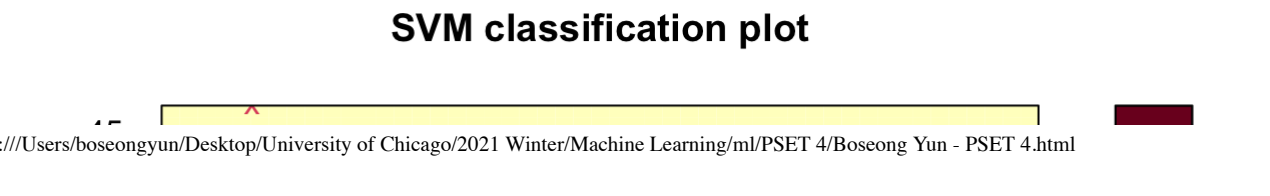

SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0

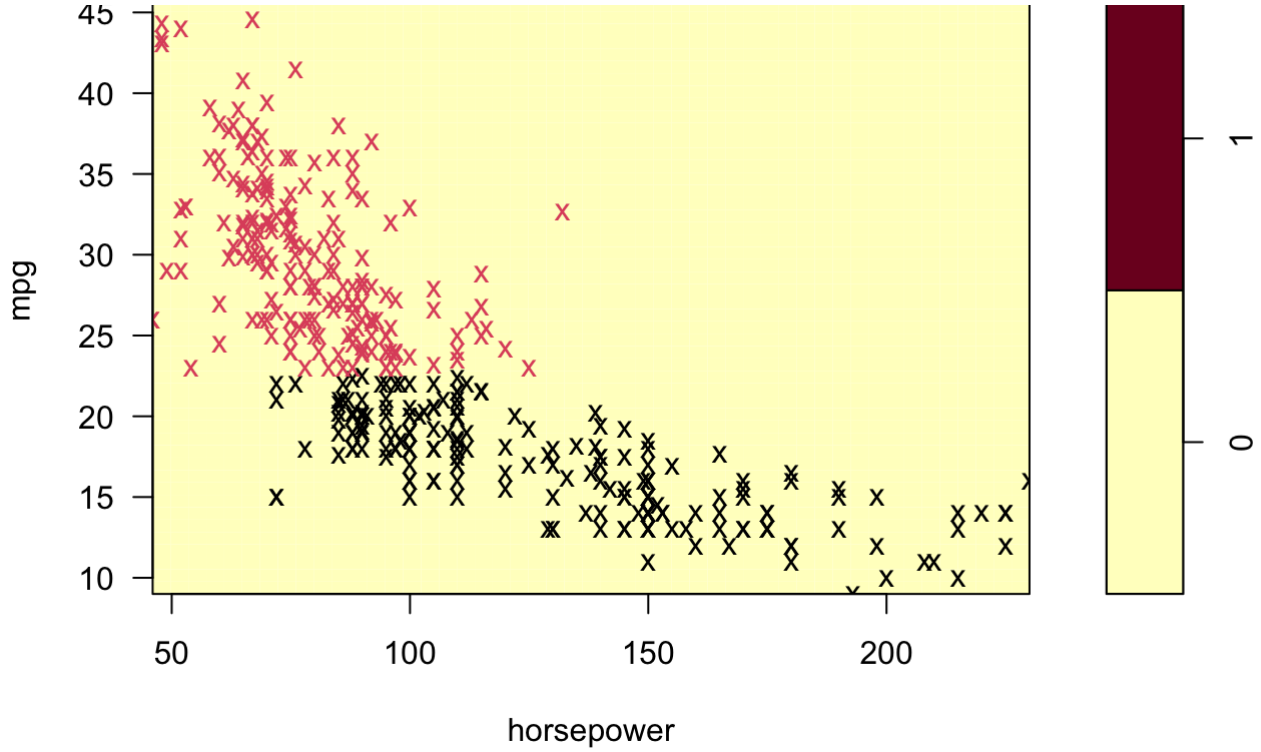



```
# SVM: polynomial kernel
plot_pair(best_polynomial)
```

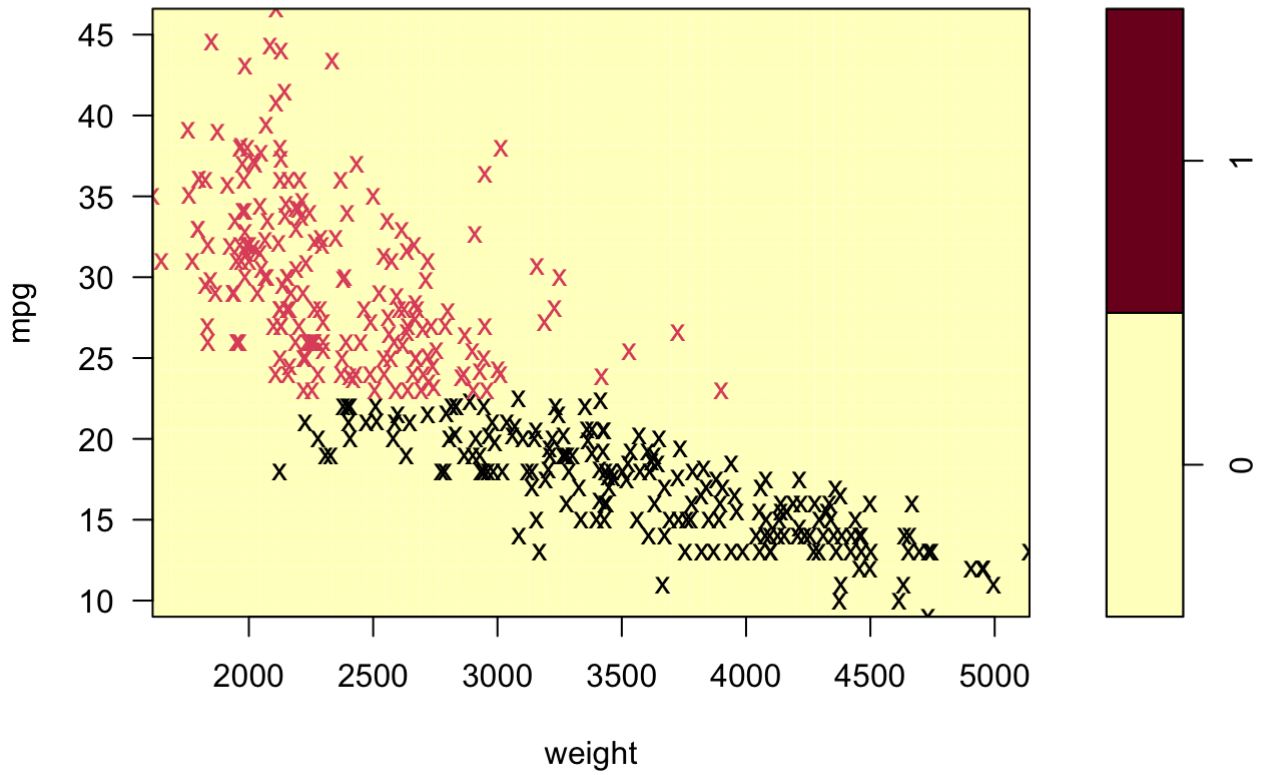

SVM classification plot
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8
1
0

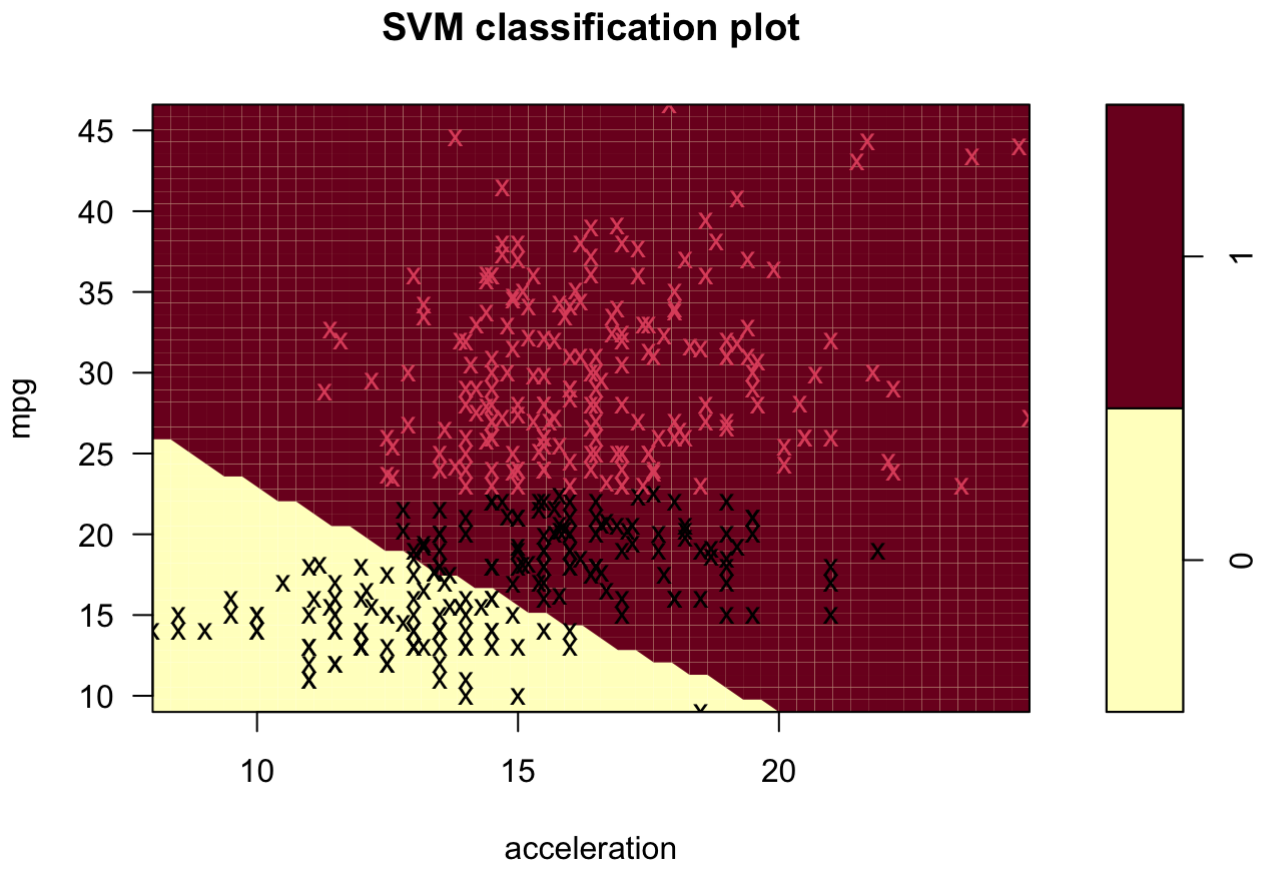

SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0

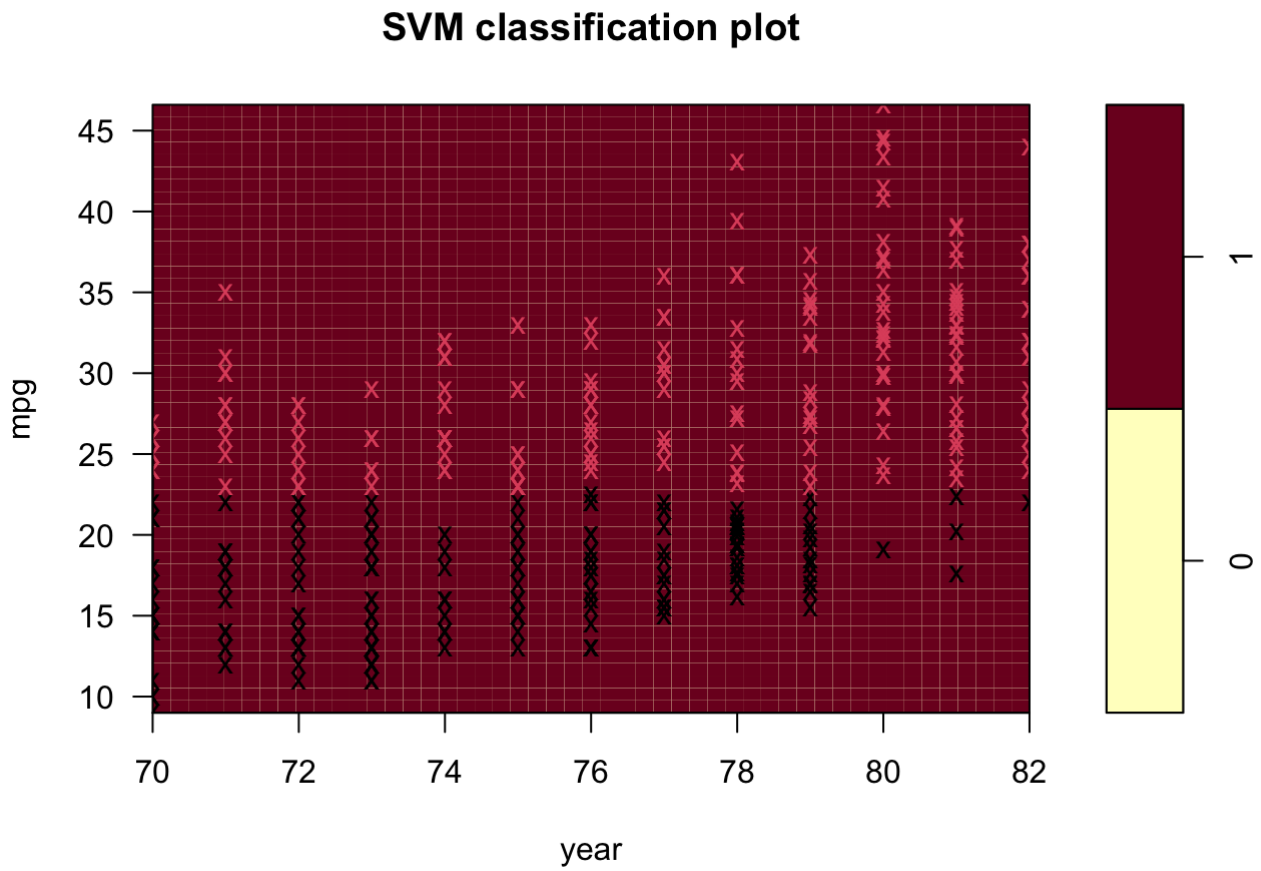

SVM classification plot

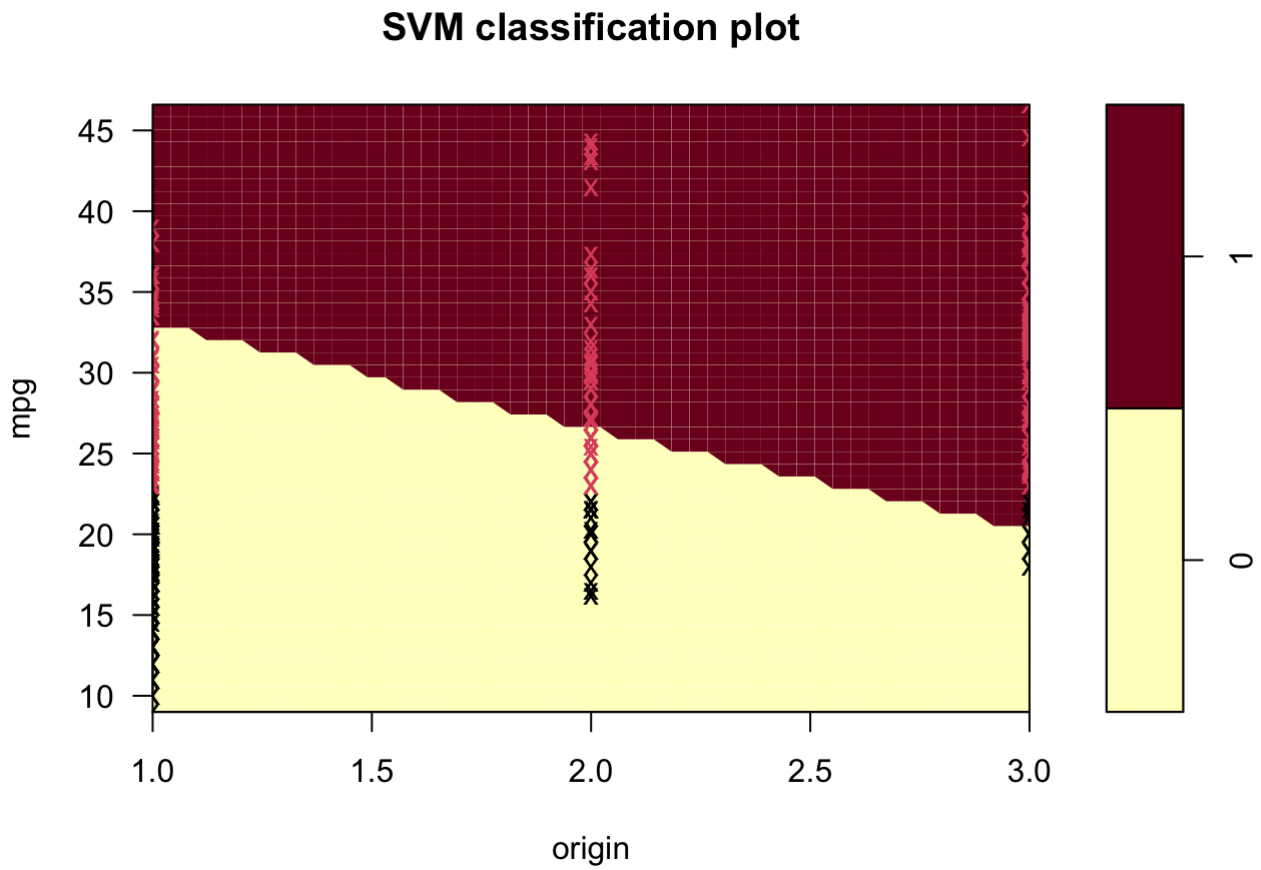


SVM classification plot




SVM classification plot
mpg
45
40
35
30
25
20
15
10
10
15
20
acceleration
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82
year
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
origin
1
0