BIOSTAT 274 Spring 2021 Homework 3

Due 11:59 PM 5/23/2020

Zian ZHUANG

Remark. For **Computational Part**, please complete your answer in the **RMarkdown** file and summit it with the generated PDF file to CCLE.

Computational Part

Q1.

(SVM, *20 pt*) 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.

```r
data(Auto)
```

(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.

Answer:

```r
df <- Auto %>%
  mutate(mpg01=ifelse(mpg > median(mpg), 1, 0) %>% as.factor) %>%
  select(-c(mpg, name))
```

(b)

Fit a support vector classifier 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.

Answer:

```r
set.seed(1996)
linear_tune <- tune(svm, mpg01 ~ ., data = df, kernel = "linear",
                    ranges = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))
summary(linear_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.08166667
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.13038462 0.07229122
## 2 1e-02 0.08692308 0.05815044
## 3 1e-01 0.08948718 0.05946716
## 4 1e+00 0.08423077 0.04195898
## 5 5e+00 0.08166667 0.03781049
## 6 1e+01 0.08166667 0.03781049
## 7 1e+02 0.08166667 0.03781049
```

```
linear_tune$best.parameters
```

```
##   cost
## 5    5
```

```
linear_tune$best.performance
```

```
## [1] 0.08166667
```

Results presents that cross-validation error was minimized when cost equals 5.

(c)

Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of `gamma` and `degree` and `cost`. Comment on your results.

Answer:

radial basis kernels:

```
set.seed(1996)
radial_tune <- tune(svm, mpg01~., data=df, kernel='radial',
                 ranges = list(cost = c(.001, .01, .1, 1, 5, 10, 100,1000),
                               gamma = c(0.5, 1, 2, 3, 4)))
radial_tune$best.parameters
```

```
##    cost gamma
## 12    1     1
```

```
radial_tune$best.performance
```

```
## [1] 0.07160256
```

As we can see from the output, the training CV error is minimized for a radial model at cost=1 and gamma=1. In addition, the training CV error is a little better than that of the linear kernel model.

polynomial basis kernels:

```
set.seed(1996)
poly_tune = tune(svm, mpg01~., data=df, kernel='polynomial',
                 ranges = list(cost = c(.001, .01, .1, 1, 5, 10, 100,1000),
                               degree = c(1,2,3,4,5)))
poly_tune$best.parameters
```

```
##    cost degree
## 21    5      3
```

```
poly_tune$best.performance
```

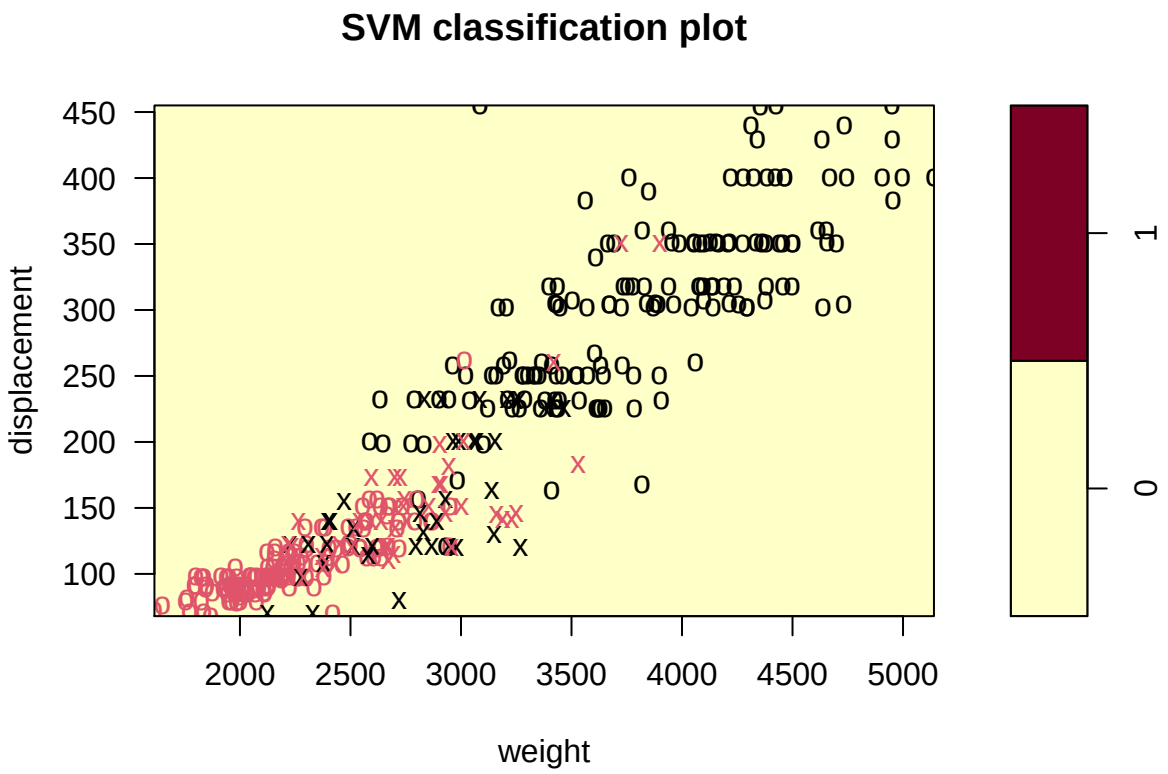
```
## [1] 0.07416667
```

As we can see from the output, the training CV error is minimized for a polynomial model at cost=5 and degree=3, which suggested the true decision boundary is non-linear. In addition, the training CV error is better than that of the linear kernel model but worse than that of the radial kernel model.

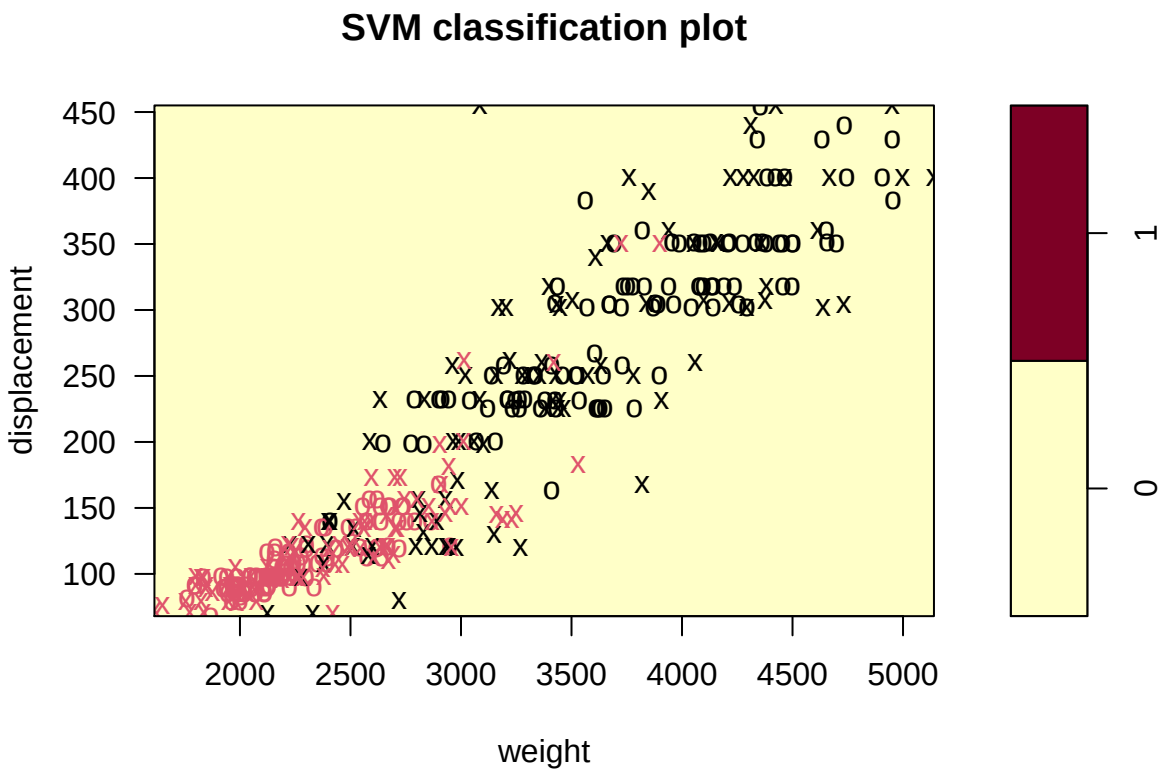
(d)

Make some plots to back up your assertions in (b) and (c).

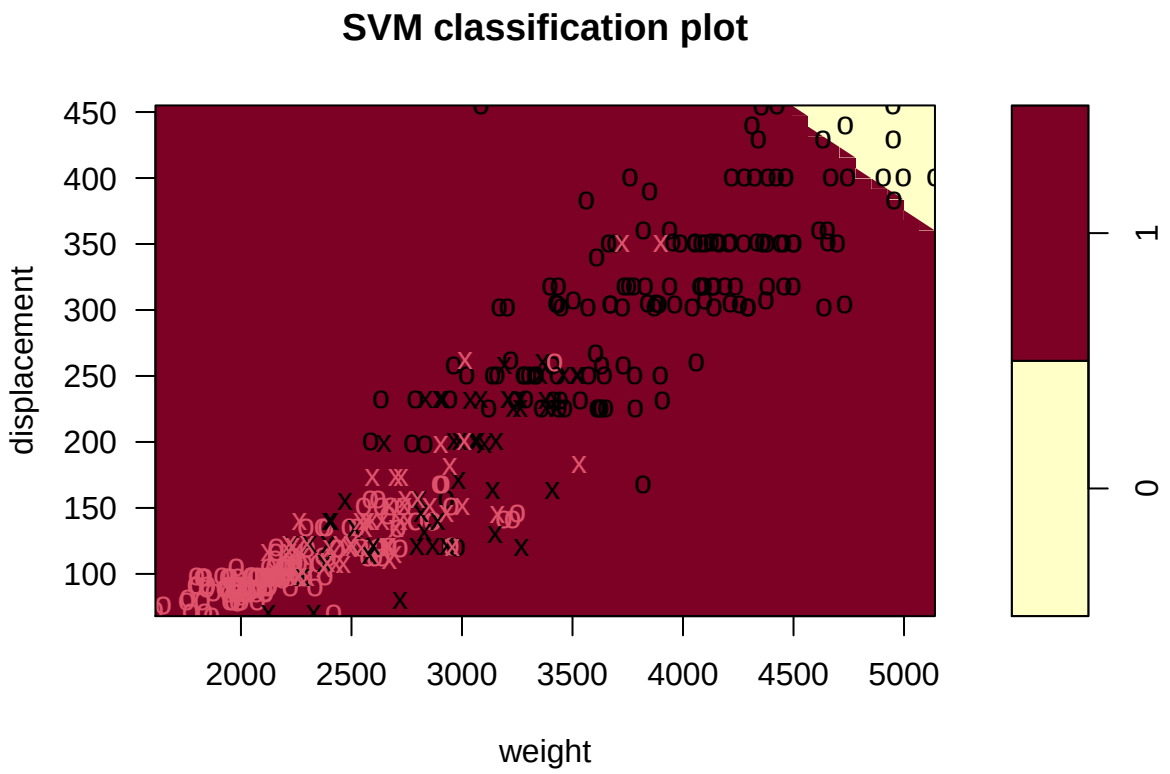
```
svmfit_l <- svm(mpg01~., data=df, kernel="linear", cost=5)
svmfit_r <- svm(mpg01~., data=df, kernel="radial", cost=1, gamma=1)
svmfit_p <- svm(mpg01~., data=df, kernel="polynomial", cost=5, degree=3)
names_list <- names(df)[-8]

# some plots

#linear
plot(svmfit_l, df, displacement~weight)
```




```
#radial
plot(svmfit_r, df, displacement~weight)
```




```
#polynomial
plot(svmfit_p, df, displacement~weight)
```



Note that more plots are put in the supplementary.

Q2.

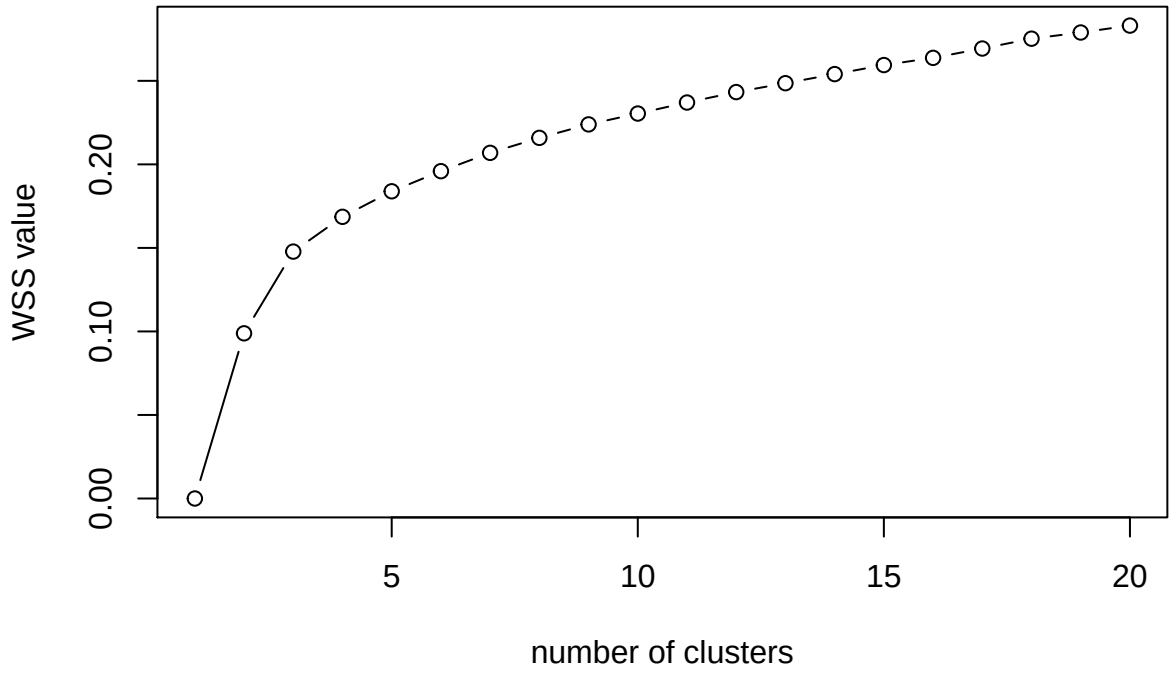
(*K*-Means Clustering, PCA and MDS, *40 pt*) The following codes read in a gene expression data from the TCGA project, which contains the expression of a random sample of 2000 genes for 563 patients from three cancer subtypes: Basal (`Basal`), Luminal A (`LumA`), and Luminal B (`LumB`). Suppose we are only interested in distinguishing Luminal A samples from Luminal B - but alas, we also have Basal samples, and we don't know which is which. Write a data analysis report to address the following problems.

```r
TCGA <- read.csv("TCGA_sample_2.txt", header = TRUE)

# Store the subtypes of tissue and the gene expression data
Subtypes <- TCGA[ ,1]
Gene <- as.matrix(TCGA[,-1])
```

(a)

Run *K*-means for K from 1 to 20 and plot the associated within cluster sum of squares (WSSs). Comment the WSS at $K = 3$.

```
wss <- NULL
for(i in 1:20){
  kmeansfit <- kmeans(Gene, centers = i, nstart = 20)
  temp <- data.frame(value=kmeansfit$betweenss/kmeansfit$totss,
                     clusters=i)
  wss <- rbind(wss, temp)
}
plot(wss$clusters, wss$value, type="b",
     xlab = "number of clusters", ylab = "WSS value")
```




```
k3 <- kmeans(Gene, centers = 3, nstart = 20)
k3$betweenss/k3$totss
```

```
## [1] 0.1477946
```

Comment the WSS at $K = 3$: The 14.8% is a measure of the total variance in our data set that is explained by the clustering. k-means minimize the within group dispersion and maximize the between-group dispersion. By assigning the samples to 3 clusters rather than 563 (number of samples) clusters achieved a reduction in sums of squares of 14.8%.

(b)

Apply K-means with $K = 3$ to the `Gene` dataset. What percentage of `Basal`, `LumA`, and `LumB` type samples are in each of the 3 resulting clusters? Did we do a good job distinguishing `LumA` from `LumB`? Confusion matrix of clusters versus subtypes might be helpful.

```
results <- data.frame(Subtypes,pred=k3$cluster) %>% filter(Subtypes!="Basal")
table(results$Subtypes,results$pred)
```

```
##
##           1   3
##   LumA 192 117
##   LumB  27 125
```

According to the simple counting table, we found that LumA should be matching with 2 and LumB should be matching with 1. Then we can have confusion matrix of clusters versus subtypes,

```
tibble(LumA = c(192, 117),
       LumB = c(27, 125),
       pred = c("LumA (pred)", "LumB (pred)")) %>%
  column_to_rownames(var = "pred")
```

```
##              LumA LumB
## LumA (pred)  192   27
## LumB (pred)  117  125
```

```
(192+125)/(461)
```

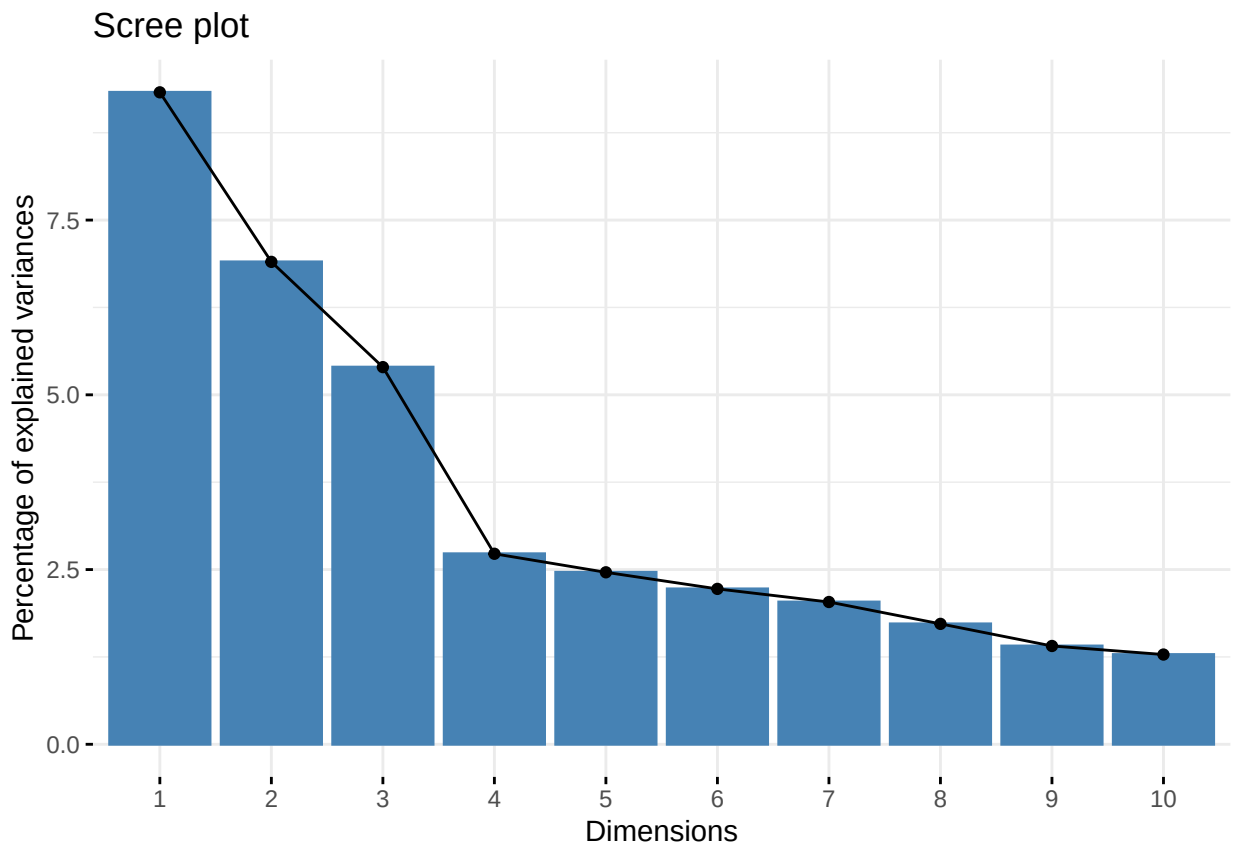
```
## [1] 0.6876356
```

As we can tell from the confusion matrix that the overall classify accuracy is 68.8%, which is satisfying.

(c)

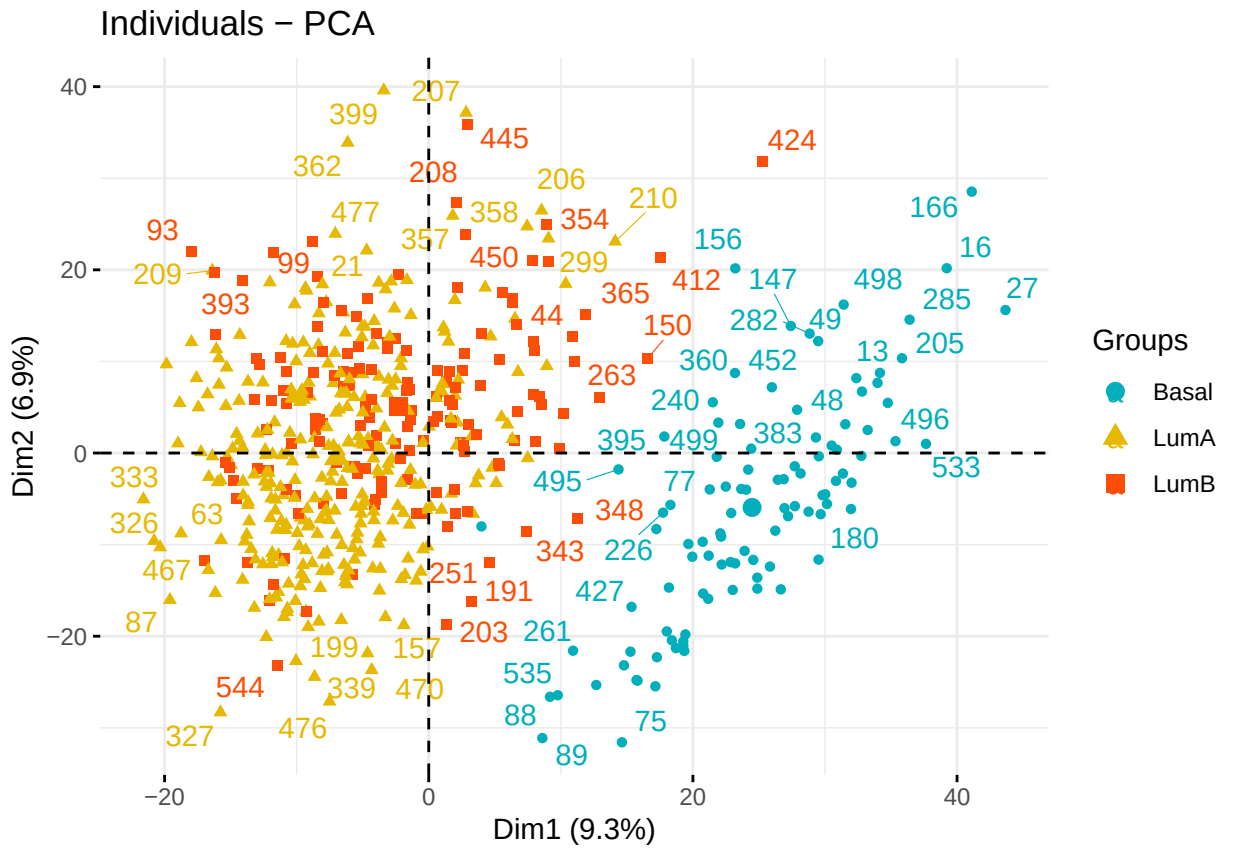
Now apply PCA to the `Gene` dataset. Plot the data in the first two PCs colored by `Subtypes`. Does this plot appear to separate the cancer subtypes well?

```
Gene_df <- Gene[, colSums(Gene == 0) != nrow(Gene)]
# Dimension reduction using PCA
res.pca <- prcomp(Gene_df,  scale = TRUE)
#Visualize eigenvalues (scree plot).
fviz_eig(res.pca)
```




```
#Graph of individuals.
fviz_pca_ind(res.pca,
             col.ind = as.factor(Subtypes), # color by groups
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             legend.title = "Groups",
             repel = TRUE
             )
```

```
## Warning: ggrepel: 492 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

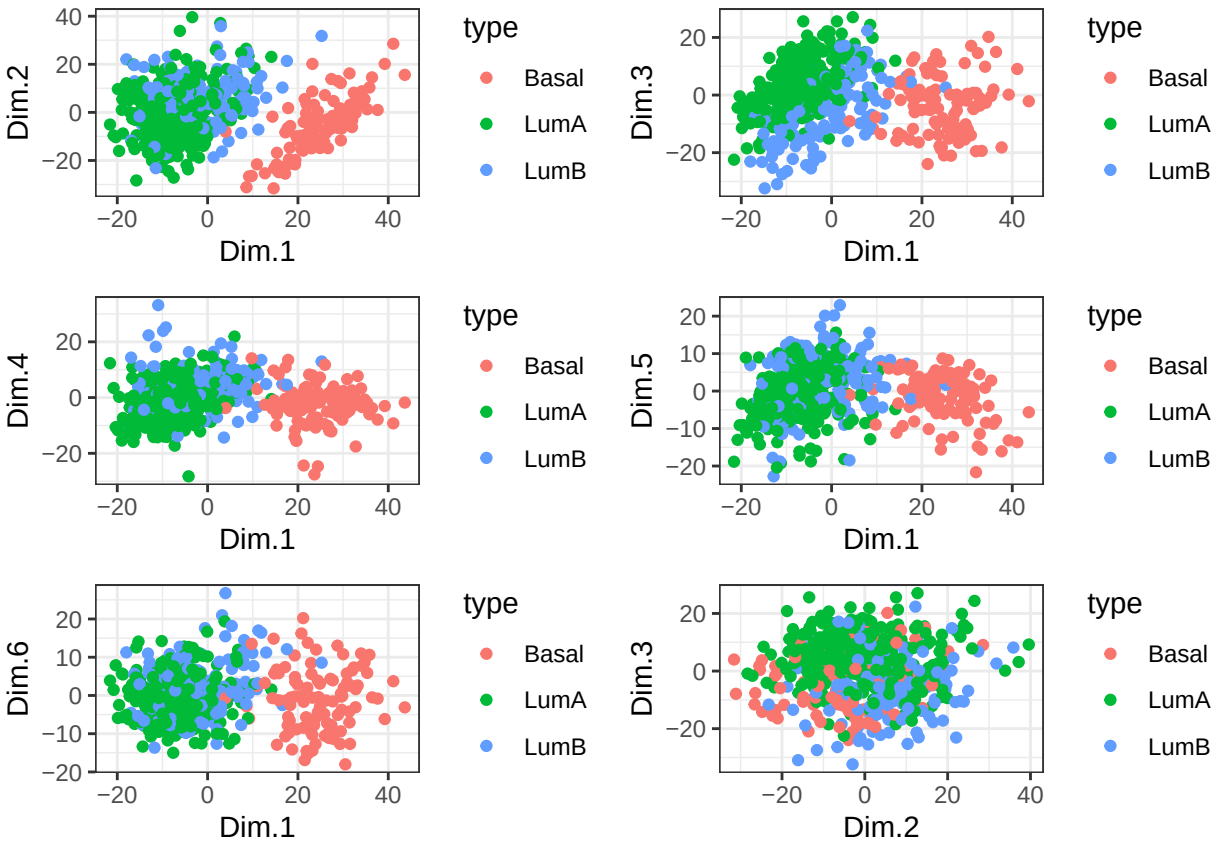


As we can tell from the plot, PCA appears to separate the cancer subtypes well.

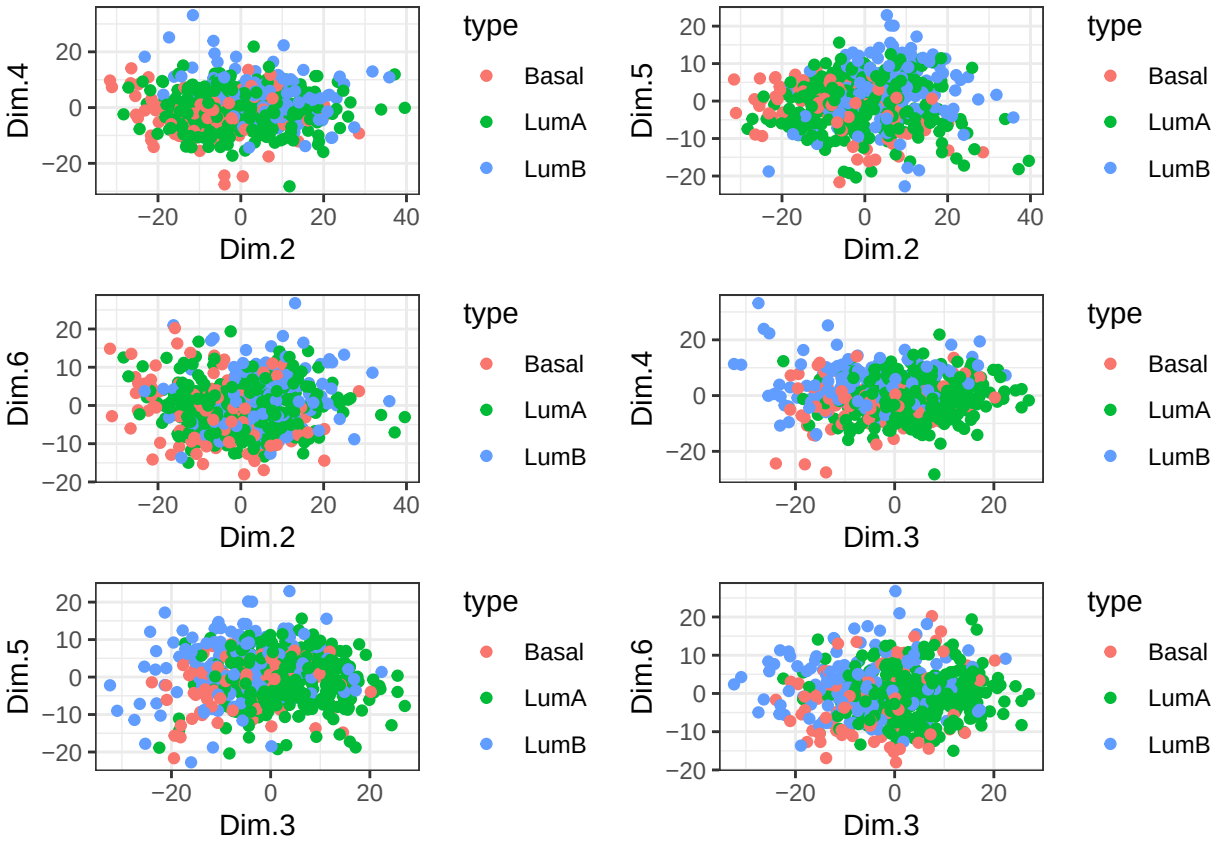
(d)

Try plotting some more PC combinations. Can you find a pair of PCs that appear to separate all three subtypes well? Report the scatterplot of the data for pair of PCs that you think best separates all three types.

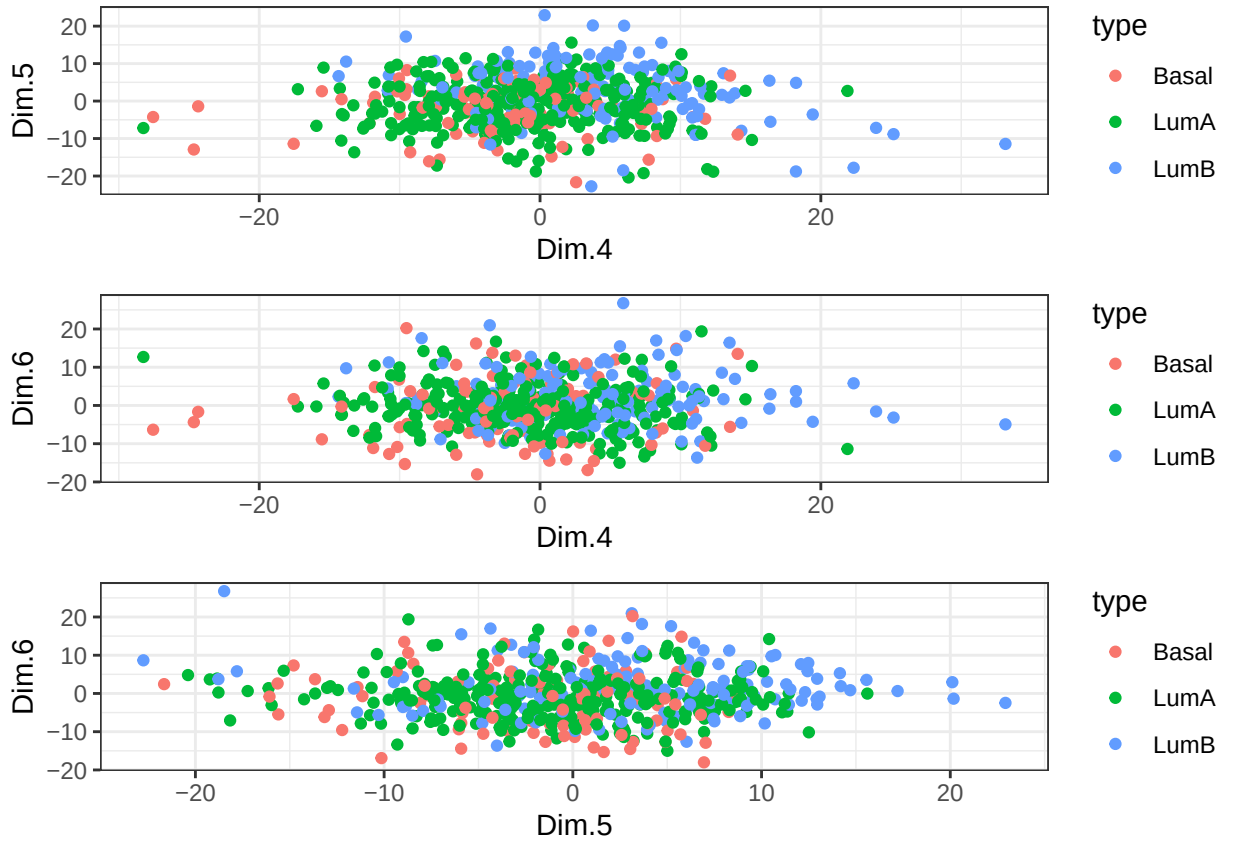
```
# Coordinates of individuals
ind.coord <- data.frame(type=as.factor(Subtypes), get_pca_ind(res.pca)$coord)

index <- 1
for(i in 1:5){
  for(k in (i+1):6){
    ggplot(ind.coord) +
      geom_point(aes_string(x=names(ind.coord)[i+1], y=names(ind.coord)[k+1],
                            color="type"))+theme_bw() -> temp
    assign(paste0("p_",index), temp)
    index <- index+1
  }
}
ggarrange(plotlist=mget(paste0("p_",c(1:6))),
          nrow = 3, ncol = 2)
```




```
ggarrange(plotlist=mget(paste0("p_",c(7:12))),
          nrow = 3, ncol = 2)
```




```
ggarrange(plotlist=mget(paste0("p_",c(13:15))),
          nrow = 3, ncol = 1)
```



According to plots, we cannot find a pair of PCs that appear to separate all three subtypes well. The best pair of PCs that separates all three types is Dim1&PCDim3.

(e)

Perform K-means with $K = 3$ on the pair of PCs identified in (d). Report the confusion matrix and make some comments.

```
k3 <- kmeans(ind.coord[,c(2,4)], centers = 3, nstart = 20)
results <- data.frame(Subtypes,pred=k3$cluster)
table(results$Subtypes,results$pred)
```

```
##
##            1   2   3
##   Basal  101   0   1
##   LumA     0 201 108
##   LumB    10  41 101
```

According to the simple counting table, we found that LumA should be matching with 2 and LumB should be matching with 3. Then we can have confusion matrix of clusters versus subtypes,

```
tibble(Basal = c(101, 1, 0),
       LumA = c(0, 201, 108),
       LumB = c(10, 41, 101),
```

```
      pred = c("Basal (pred)", "LumA (pred)", "LumB (pred)")) %>%
  column_to_rownames(var = "pred")
```

```
##              Basal LumA LumB
## Basal (pred)   101    0   10
## LumA (pred)      1  201   41
## LumB (pred)      0  108  101
```

```
(201+101)/(201+108+41+101)
```

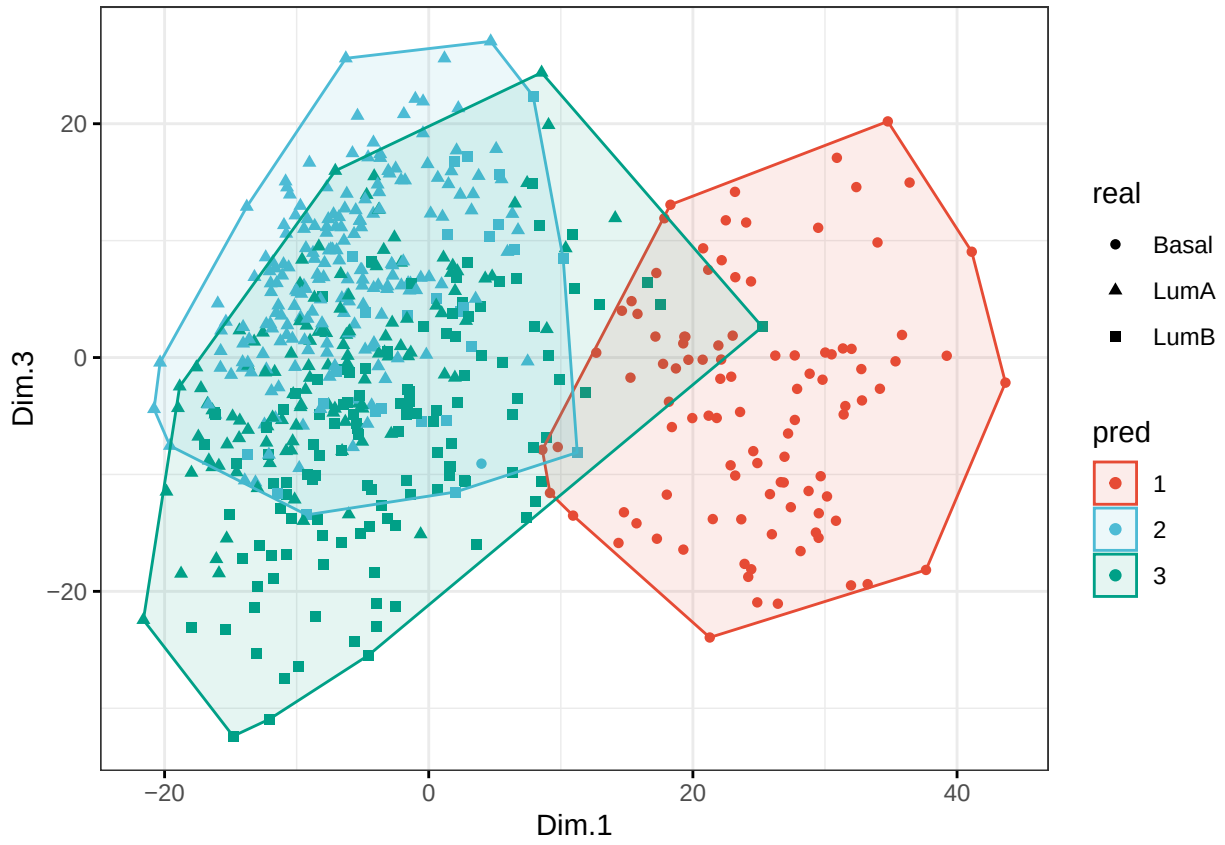
```
## [1] 0.6696231
```

As we can tell from the confusion matrix for `LumA` and `LumB` that the overall classify accuracy is 67.0%, which is largely consistent with the results when we applied K-means with $K = 3$ algorithm directly.

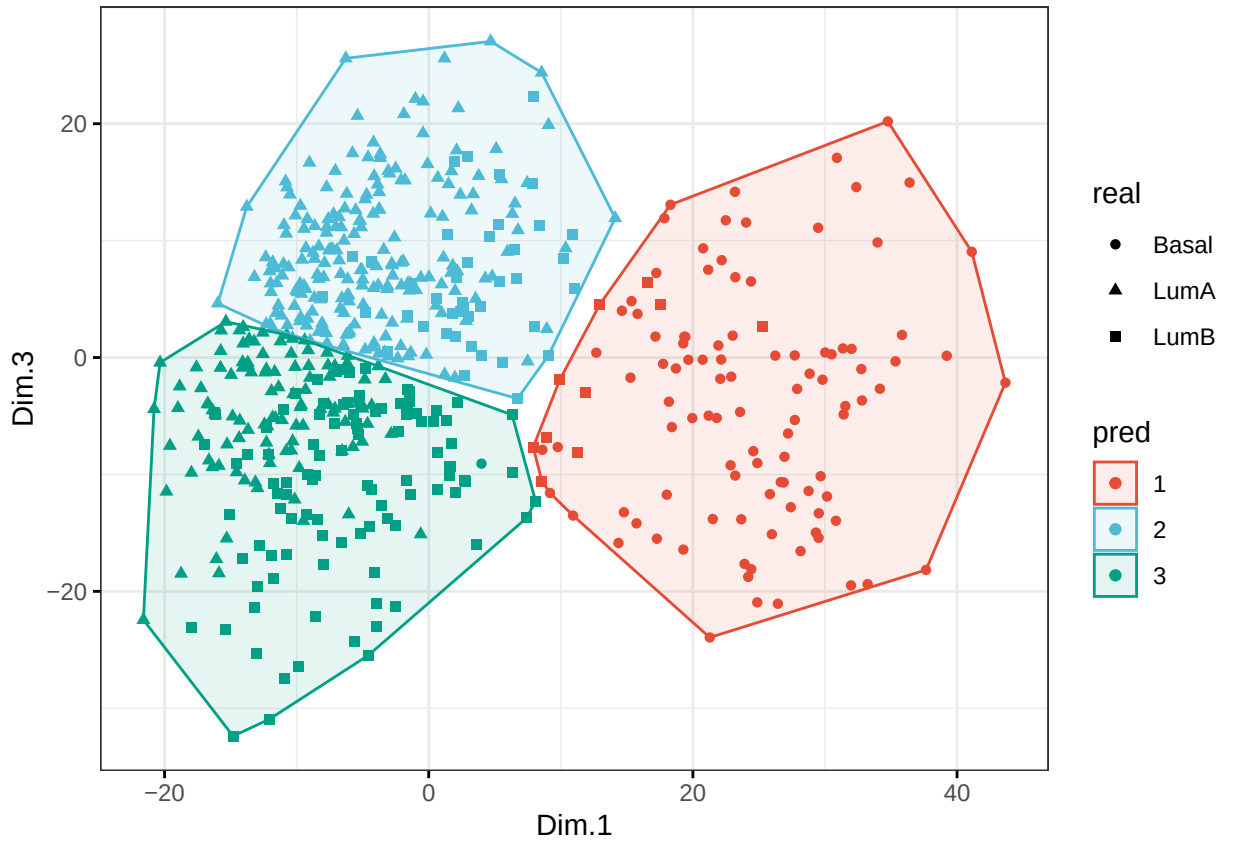
(f)

Create two plots colored by the clusters found in (b) and in (e) respectively. Do they look similarly or differently? Explain why using PCA to reduce the number of dimensions from 2000 to 2 did not significantly change the results of K-means.

```
k3_origin <- kmeans(Gene_df, centers = 3, nstart = 20)
# clusters found in (b)
data.frame(real=as.factor(Subtypes),
           pred=as.factor(k3_origin$cluster),ind.coord)%>%
  ggscatter(
  ., x = "Dim.1", y = "Dim.3",
  color = "pred", palette = "npg", ellipse = TRUE, ellipse.type = "convex",
  shape = "real", size = 1.5,  legend = "right", ggtheme = theme_bw()
)
```




```
# clusters found in (e)
data.frame(real=as.factor(Subtypes),
           pred=as.factor(k3$cluster),ind.coord)%>%
  ggscatter(
  ., x = "Dim.1", y = "Dim.3",
  color = "pred", palette = "npg", ellipse = TRUE, ellipse.type = "convex",
  shape = "real", size = 1.5,  legend = "right", ggtheme = theme_bw()
)
```



As we can tell from the plots that two model provided different clusters (but similar classification accuracy). Research (Yeung & Ruzzo, 2000) showed that clustering with the PC's rather than the original dims does not necessarily improve cluster quality. In this question, since none of PC's (which contain most of the variation in the data) capture the cluster structure well, it cannot improve the classification accuracy.

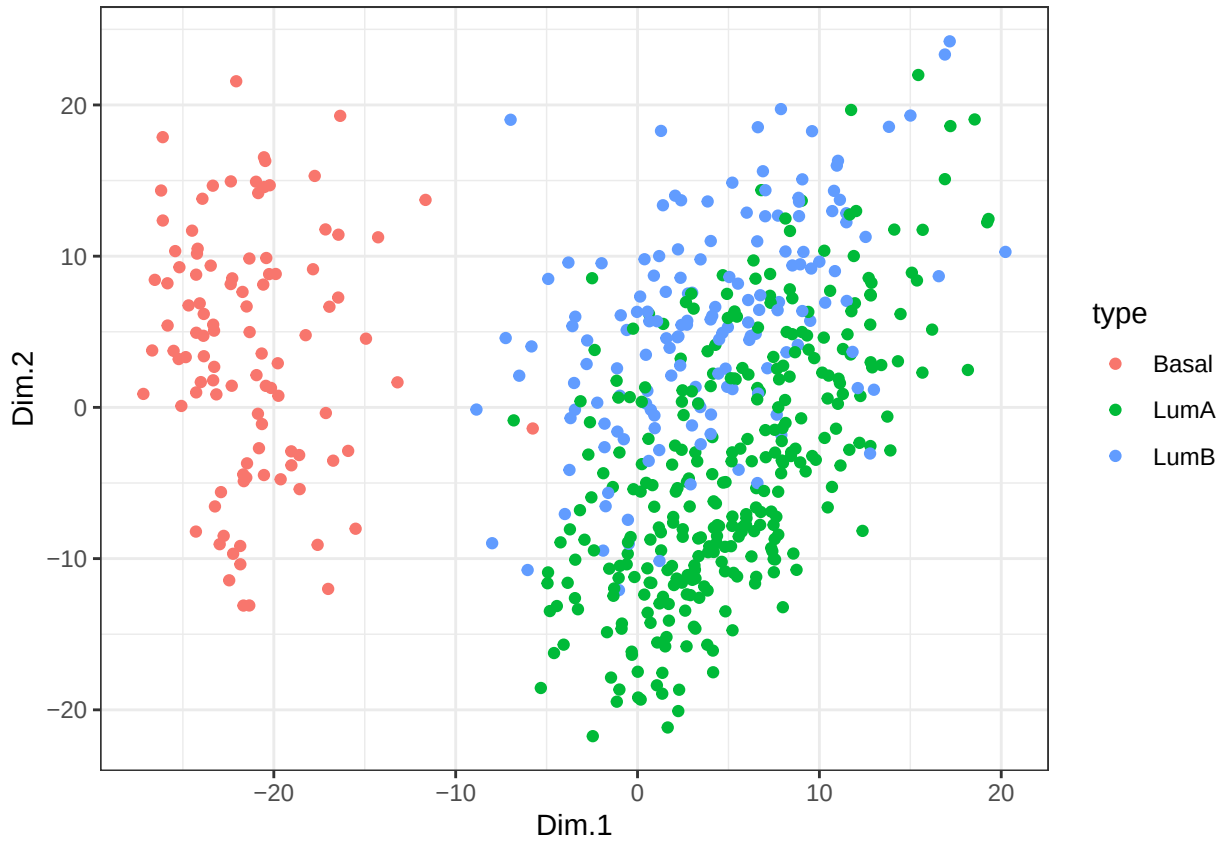
(g)

Now apply MDS with various metrics and non-metric MDS to `Gene` to obtain 2-dimensional representations. Does any of them provide better separated scatterplot as compared to that from (d)? Notice that the Euclidean metric in MDS gives the same representation as PCA does.

```
# Compute classic MDS
mds_c <- Gene %>%
  dist() %>%
  cmdscale(k = 2) %>%
  as_tibble() %>%
  mutate(type=as.factor(Subtypes))
```

```
## Warning: The ‘x‘ argument of ‘as_tibble.matrix()‘ must have unique column names if ‘.name_repair‘ is
## Using compatibility ‘.name_repair‘.
```

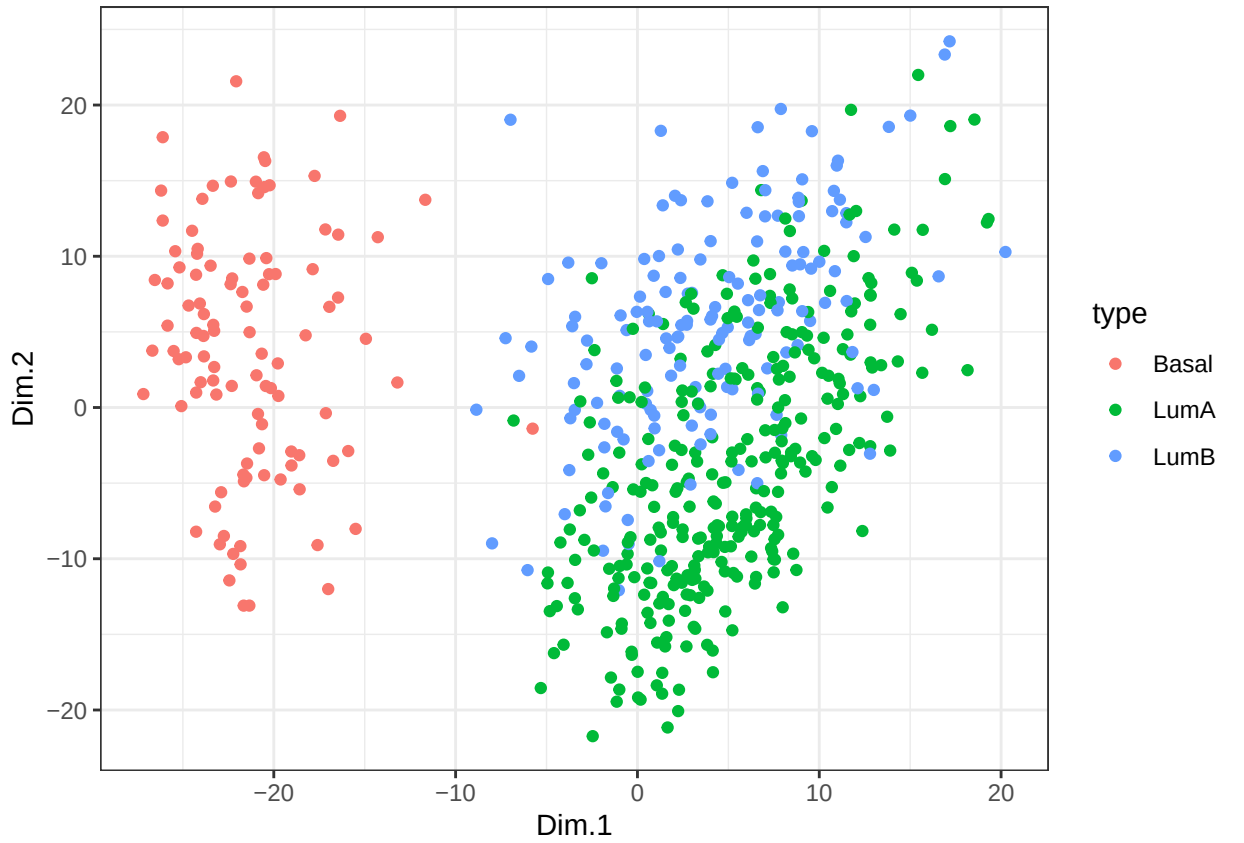
```
colnames(mds_c)[1:2] <- c("Dim.1", "Dim.2")
# Plot MDS
ggplot(mds_c) +
  geom_point(aes(x=Dim.1, y=Dim.2, color=type)) + theme_bw()
```




```
# Compute non-metric MDS
mds_n <- Gene %>%
  dist() %>%
  isoMDS(k = 2) %>%
  .$points %>%
  as_tibble() %>%
  mutate(type=as.factor(Subtypes))
```

```
## initial  value 33.268145
## final  value 33.250852
## converged
```

```
colnames(mds_n)[1:2] <- c("Dim.1", "Dim.2")
# Plot MDS
ggplot(mds_n) +
  geom_point(aes(x=Dim.1, y=Dim.2, color=type)) + theme_bw()
```



We can tell from the plots that both of them provide similar separated scatterplot as compared to that from (d).

(h)

Perform K-means with $K = 3$ on the new representations from (g) and report the confusion matrices. Compare them with that from (e).

```
k3_mdsc <- kmeans(mds_c[,1:2], centers = 3, nstart = 20)
results <- data.frame(Subtypes,pred=k3_mdsc$cluster)
table(results$Subtypes,results$pred)
```

```
##
##            1   2   3
##   Basal  101   1   0
##   LumA     0 195 114
##   LumB     2  33 117
```

```
k3_mdsn <- kmeans(mds_n[,1:2], centers = 3, nstart = 20)
results <- data.frame(Subtypes,pred=k3_mdsn$cluster)
table(results$Subtypes,results$pred)
```

```
##
##            1   2   3
```

```
##   Basal 101   0   1
##   LumA    0 114 195
##   LumB    2 117  33
```

According to the simple counting table, we found that classic mds and non-metric mds provides same results. Then we can have confusion matrix of clusters versus subtypes,

```
tibble(LumA = c(195, 114),
       LumB = c(33, 117),
       pred = c( "LumA (pred)", "LumB (pred)")) %>%
  column_to_rownames(var = "pred")
```

```
##             LumA LumB
## LumA (pred)  195   33
## LumB (pred)  114  117
```

```
(195+117)/(195+114+33+117)
```

```
## [1] 0.6797386
```

As we can tell from the confusion matrix for `LumA` and `LumB` that the overall classify accuracy is 68.0%, which is very close to the results we obtained in (e).

(i)

Suppose we might know that the first PC contains information we aren't interested in. Apply K-means with $K = 3$ to `Gene` dataset **subtracting the approximation from the first PC**. Report the confusion matrix and make some comments.

```
k3 <- kmeans(ind.coord[,-c(1:2)], centers = 3, nstart = 20)
results <- data.frame(Subtypes,pred=k3$cluster)
table(results$Subtypes,results$pred)
```

```
##
##            1   2   3
##   Basal   36  52  14
##   LumA    61 140 108
##   LumB    94  19  39
```

According to the simple counting table, we found that classic mds and non-metric mds provides same results. Then we can have confusion matrix of clusters versus subtypes,

```
tibble(LumA = c(140, 61),
       LumB = c(19, 94),
       pred = c( "LumA (pred)", "LumB (pred)")) %>%
  column_to_rownames(var = "pred")
```

```
##             LumA LumB
## LumA (pred)  140   19
## LumB (pred)   61   94
```

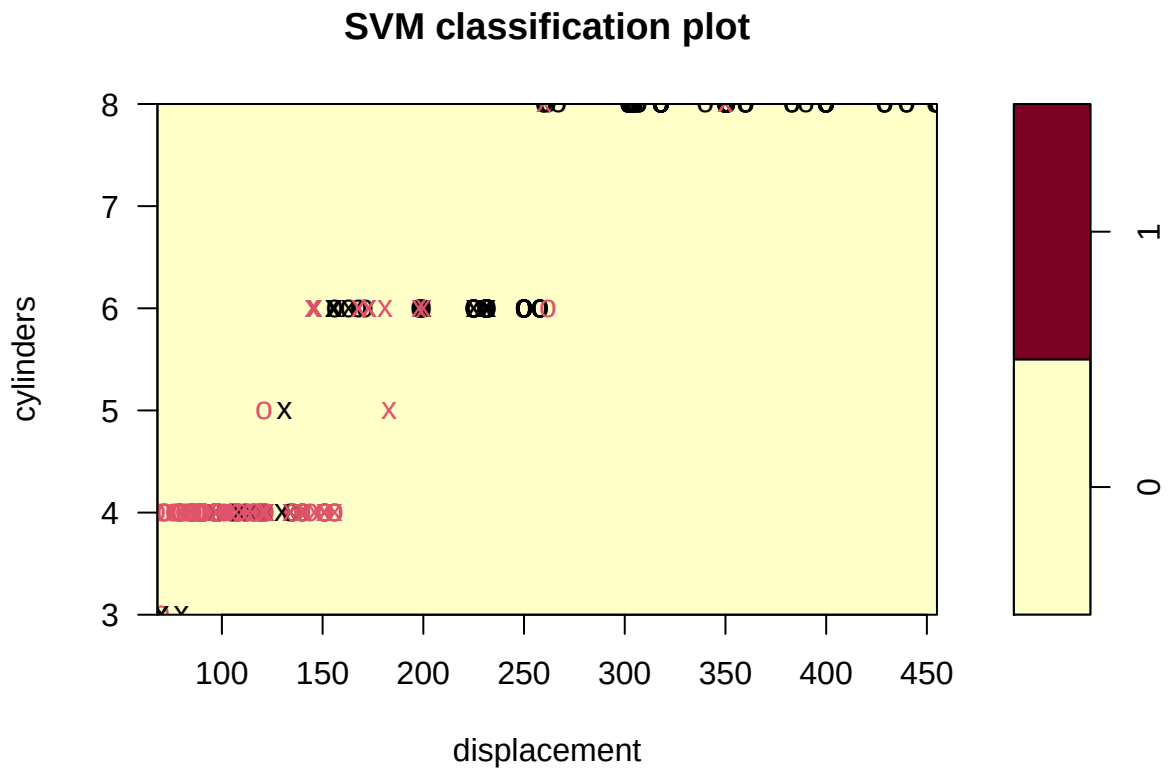
```r
(140+94)/(140+61+19+94)
```

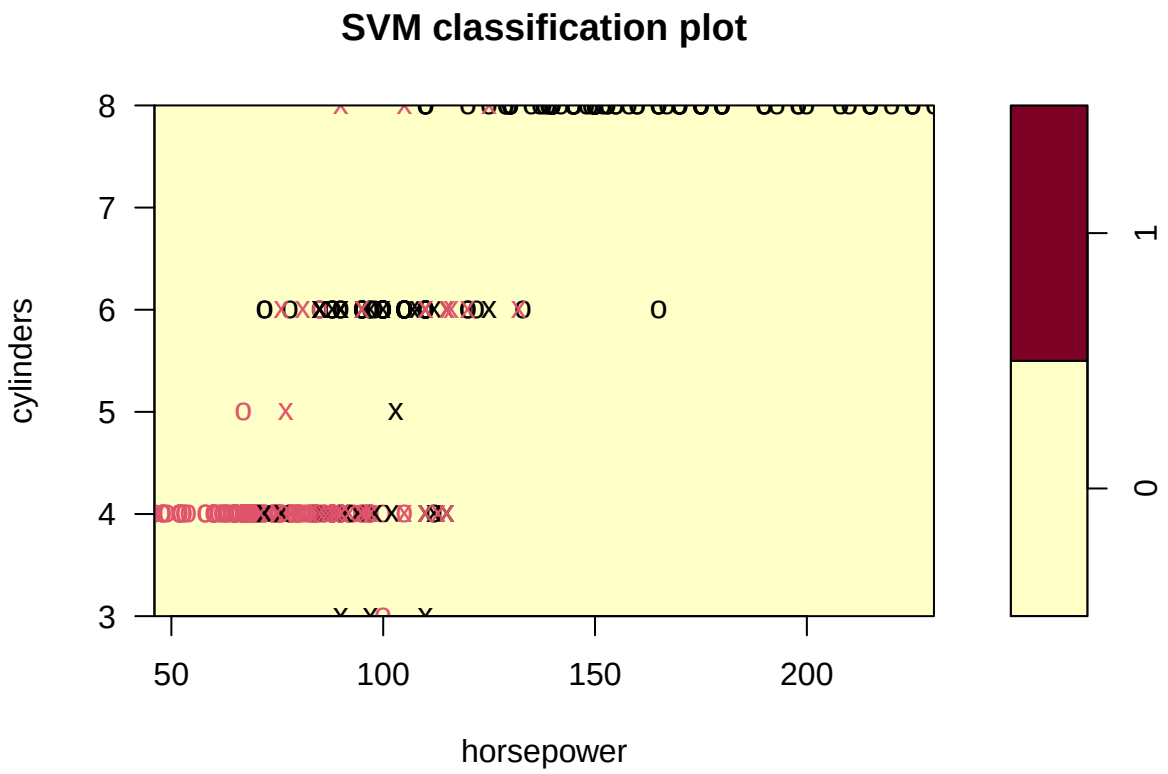
```
## [1] 0.7452229
```

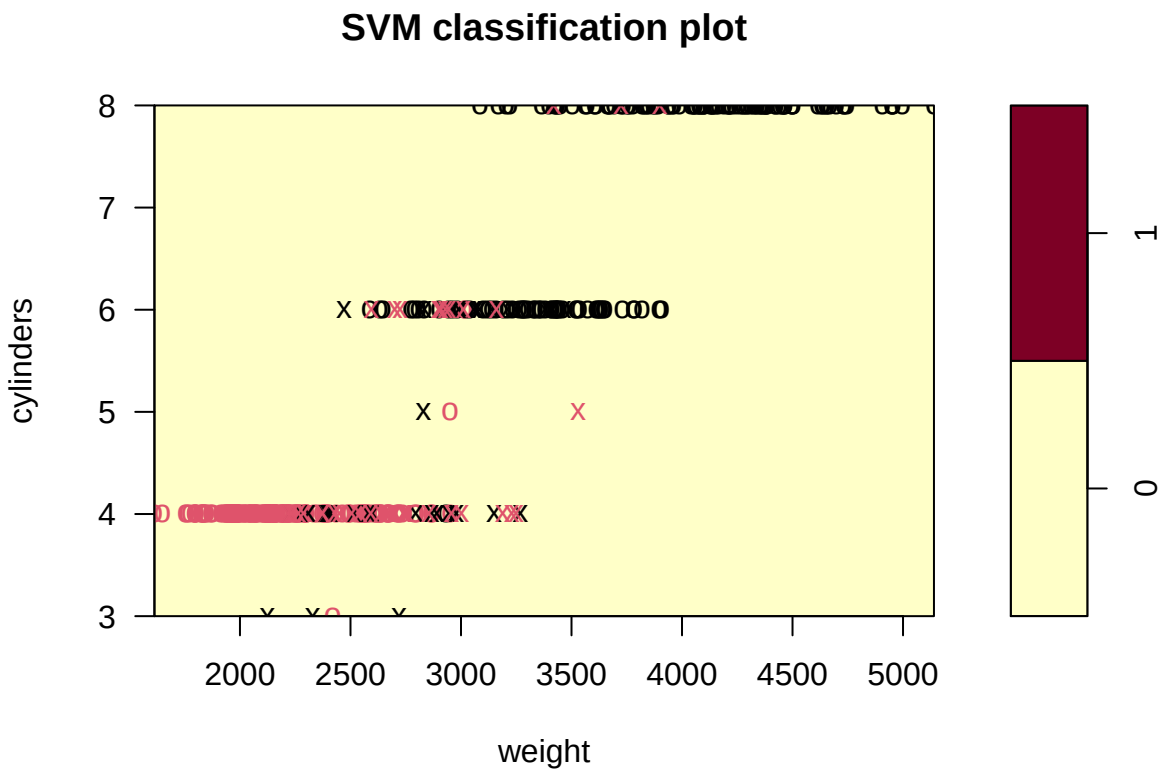
As we can tell from the confusion matrix for `LumA` and `LumB` that the overall classify accuracy is 74.5%, which is better than the model including the first PC. But 106 `LumA` and 39 `LumB` cases are wrongly classified as `Basal`, which is much worse than the model including the first PC ($\leqslant$ 10 `LumA` and $\leqslant$ 10 `LumB`).

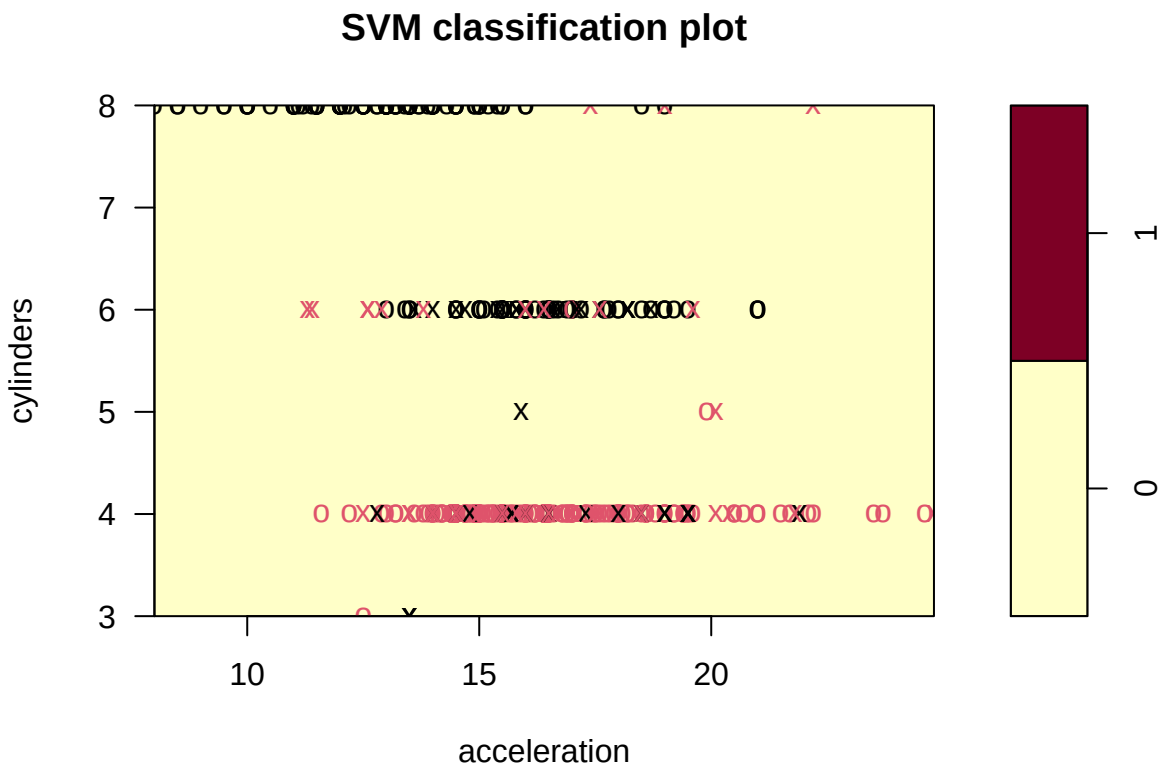
Supplementary

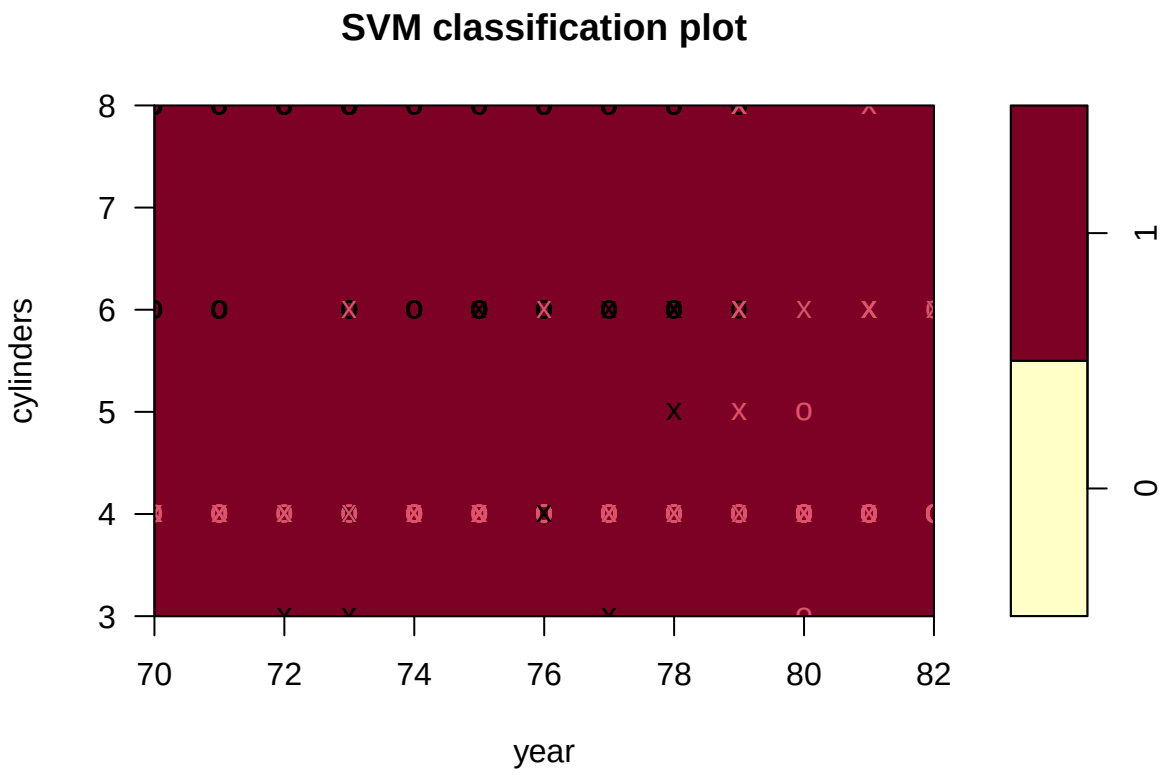
```r
#linear
for(i in 1:2){
  for(k in (i+1):7){
    plot(svmfit_l, df,
         as.formula(paste0(names_list[i], "~", names_list[k])))
  }
}
```

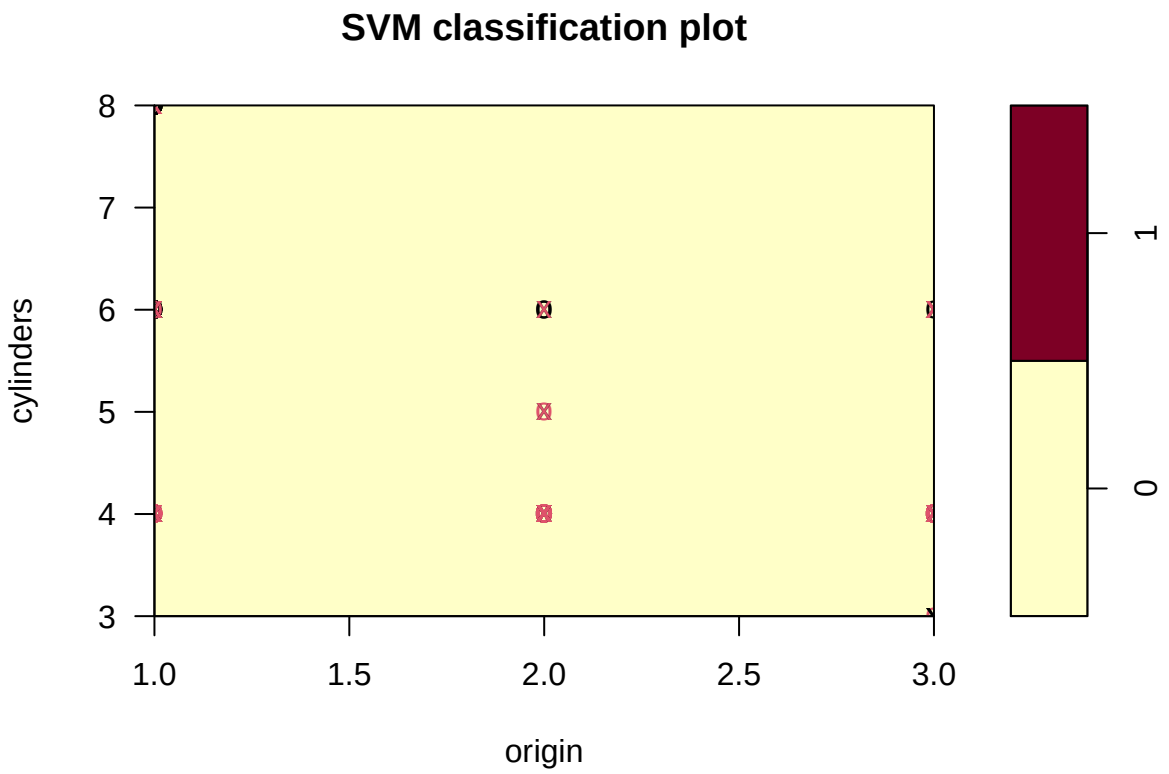


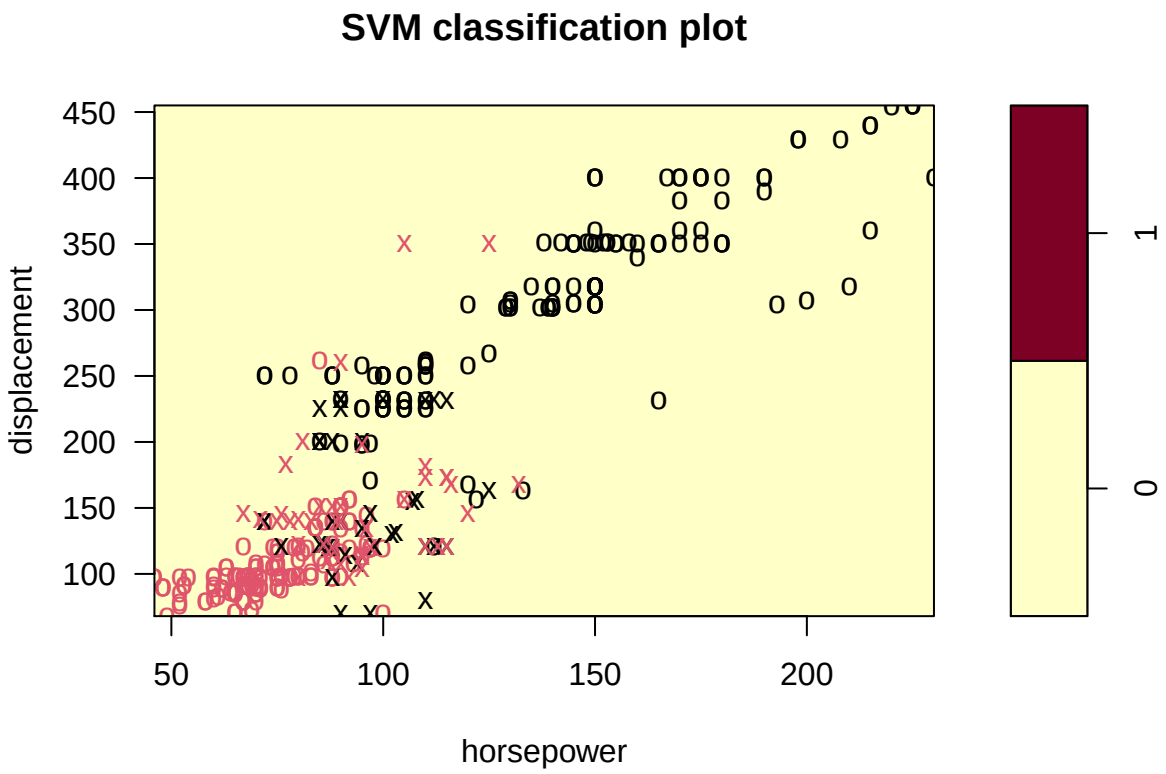

SVM classification plot
cylinders
horsepower
8
7
6
5
4
3
50
100
150
200
1
0

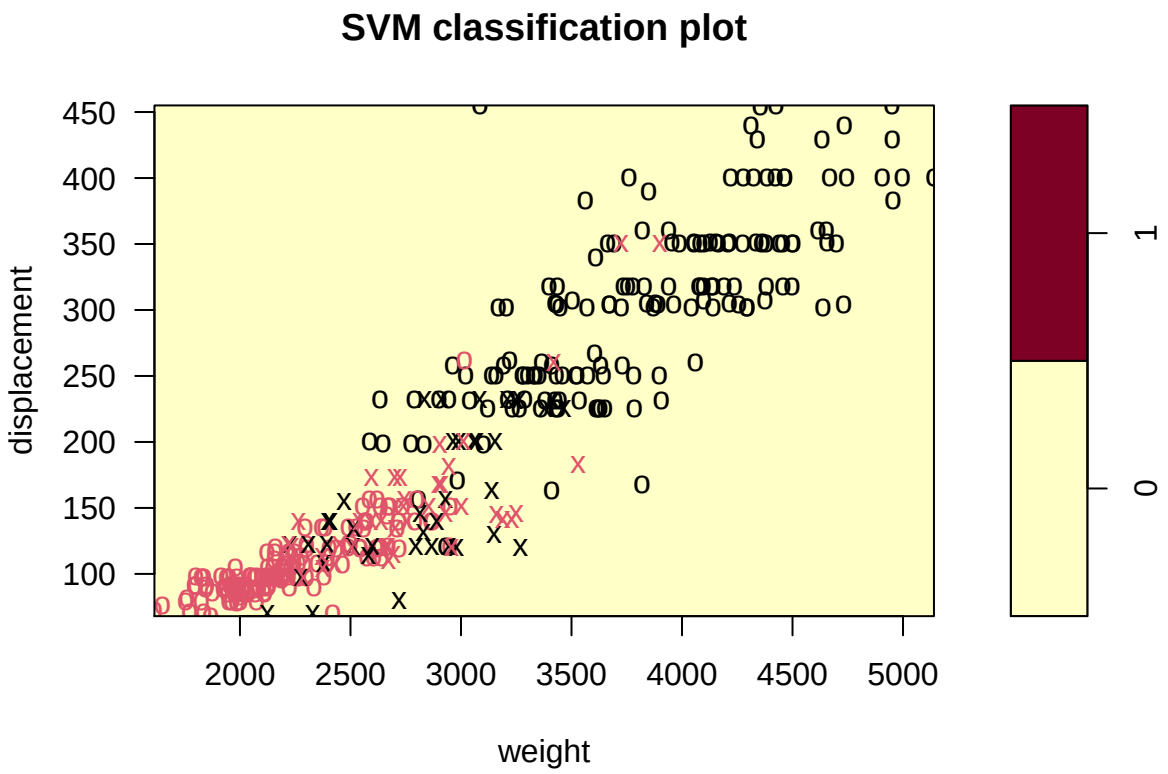

SVM classification plot
cylinders
weight
8
7
6
5
4
3
2000
2500
3000
3500
4000
4500
5000
1
0

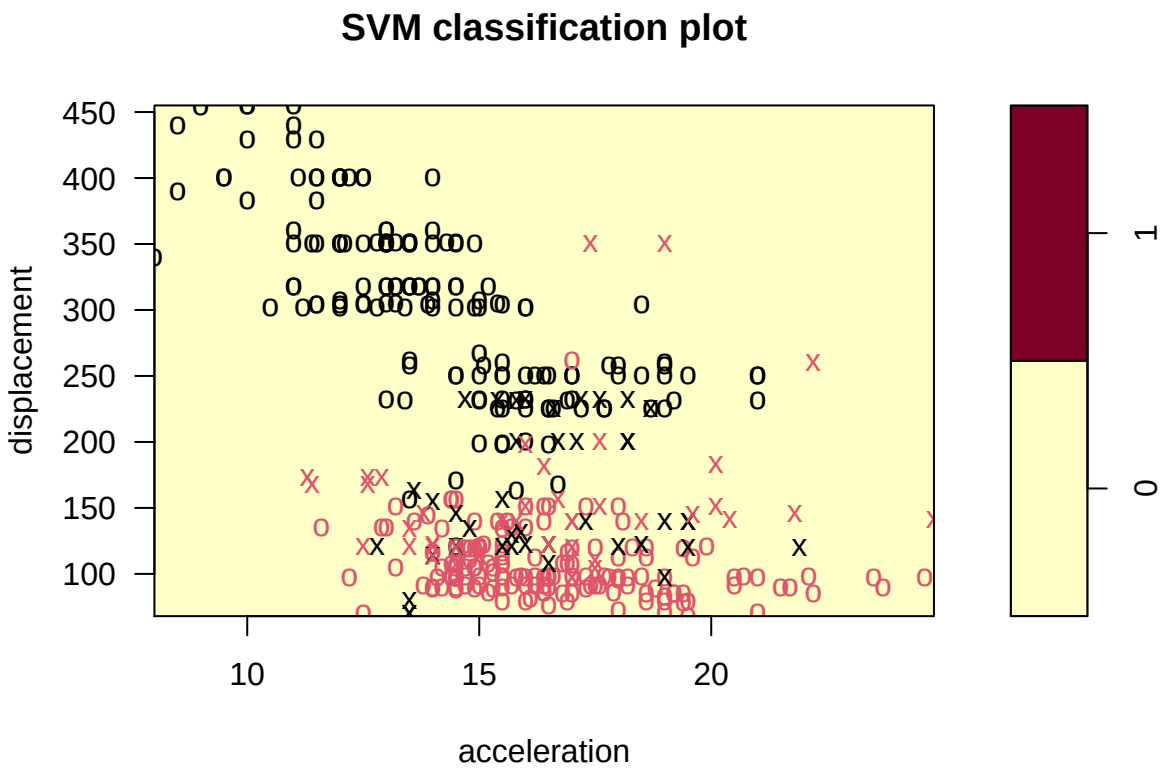

SVM classification plot
cylinders
acceleration
8
7
6
5
4
3
10
15
20
1
0

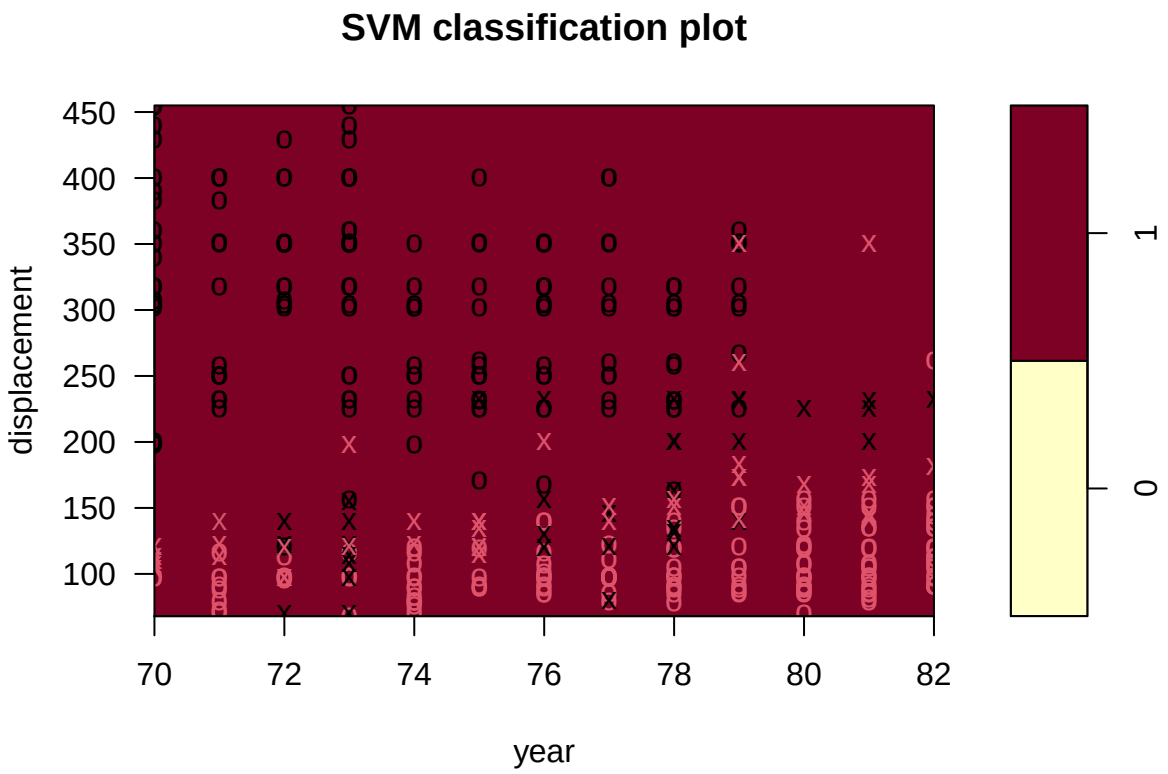

SVM classification plot
cylinders
year
8
7
6
5
4
3
70
72
74
76
78
80
82
1
0

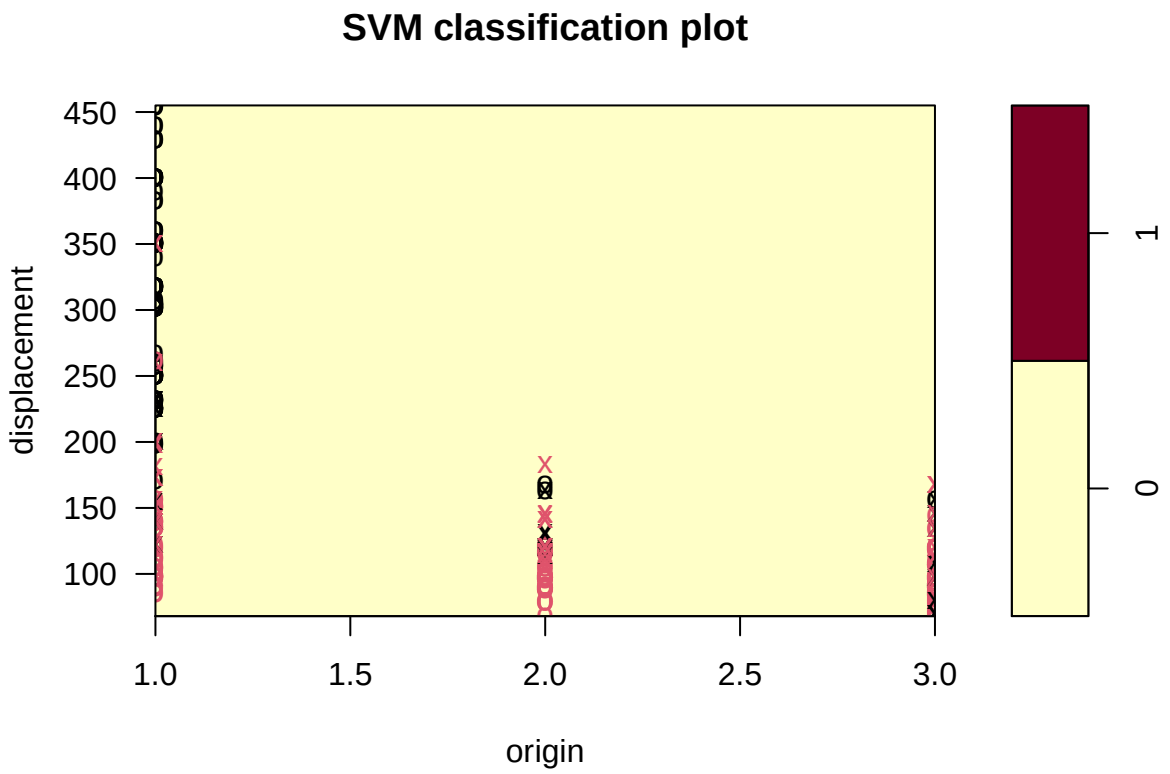

SVM classification plot
cylinders
origin
8
7
6
5
4
3
1.0
1.5
2.0
2.5
3.0
1
0


SVM classification plot
displacement
horsepower
450
400
350
300
250
200
150
100
50
100
150
200
1
0

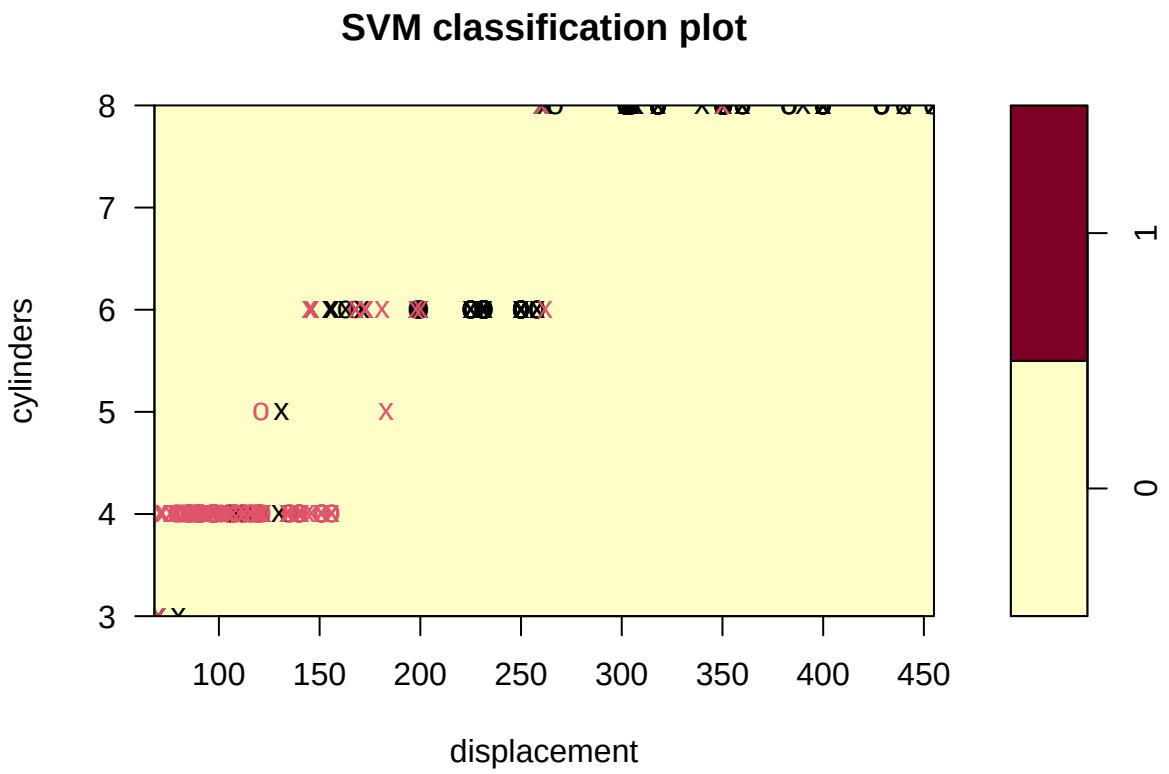

SVM classification plot
displacement
weight
450
400
350
300
250
200
150
100
2000
2500
3000
3500
4000
4500
5000
1
0

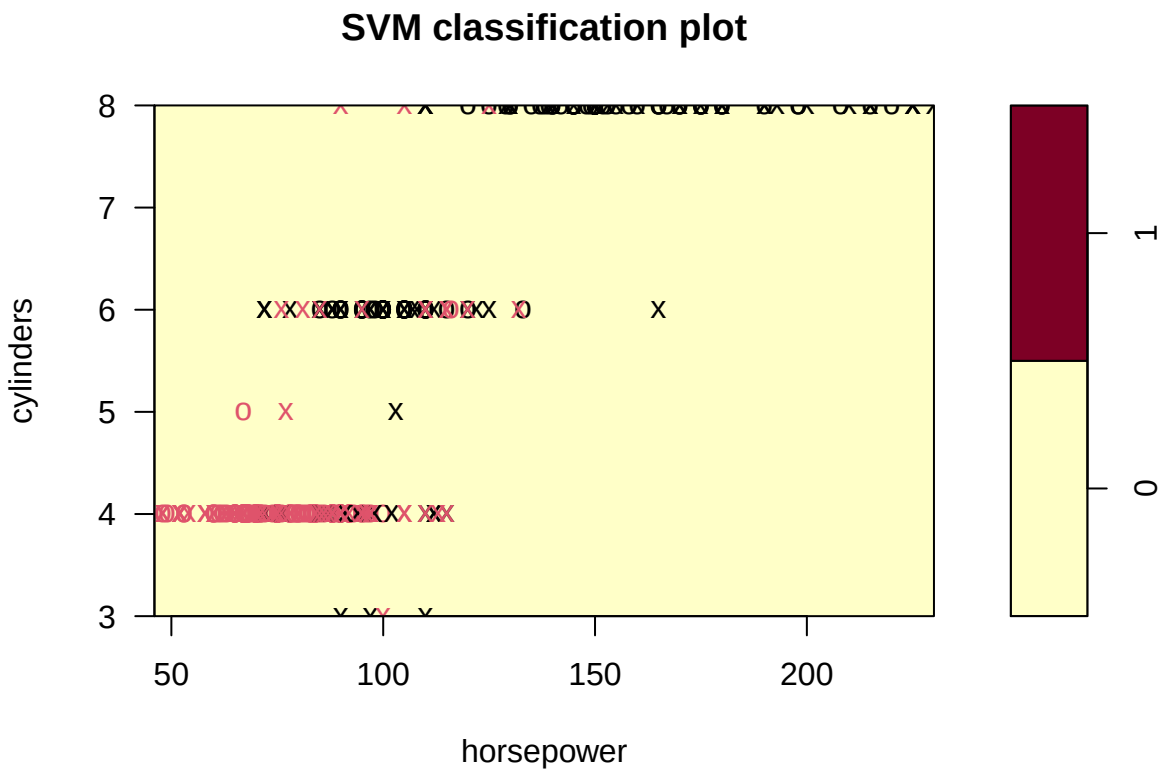

SVM classification plot
displacement
acceleration
450
400
350
300
250
200
150
100
10
15
20
1
0

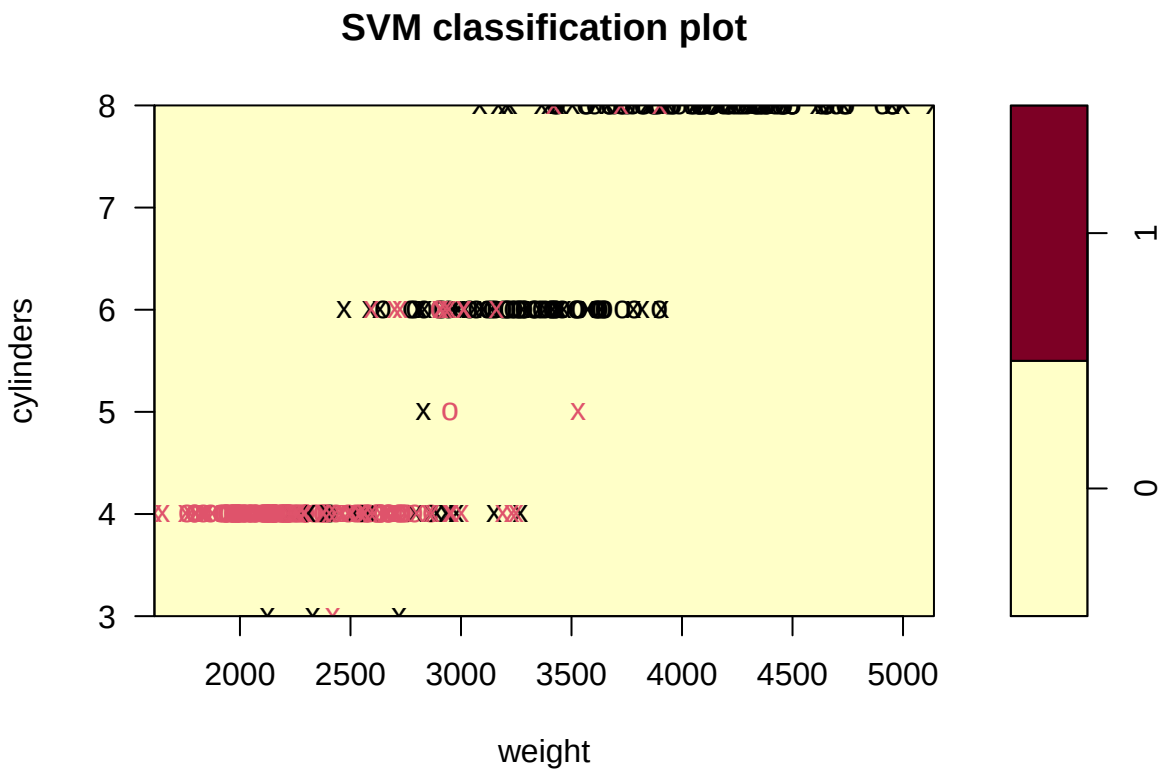

SVM classification plot
displacement
450
400
350
300
250
200
150
100
70
72
74
76
78
80
82
year
1
0

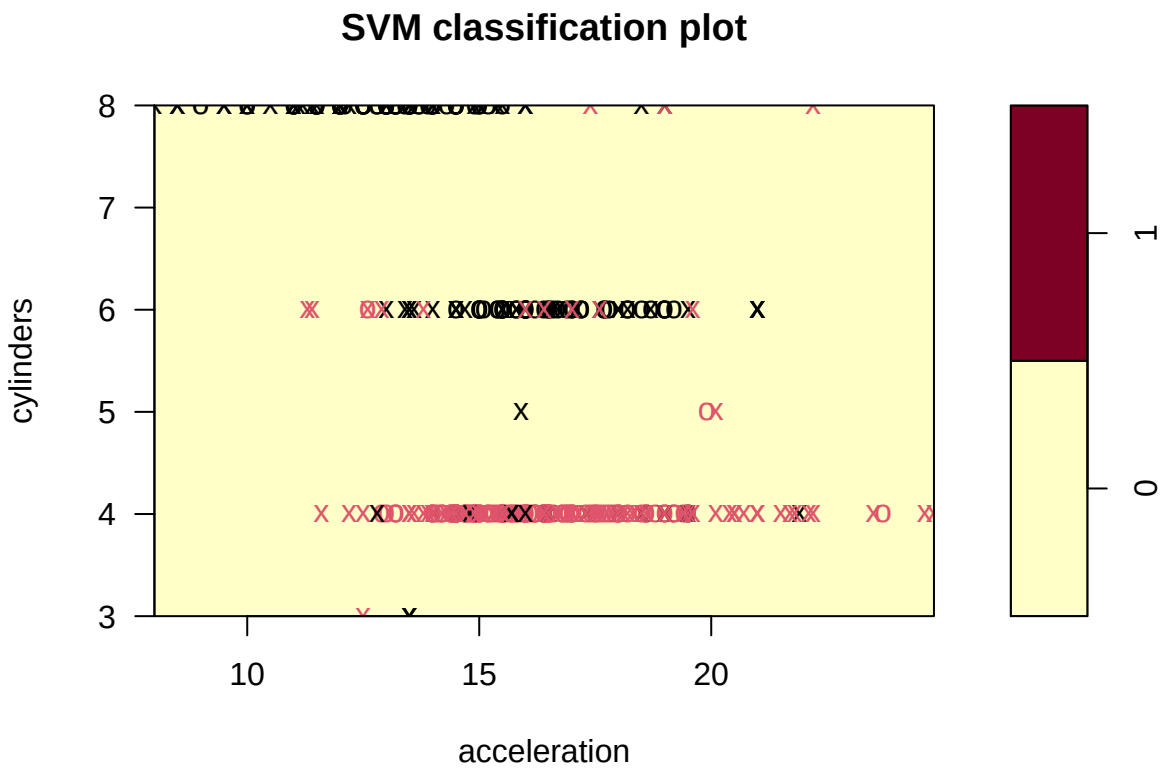


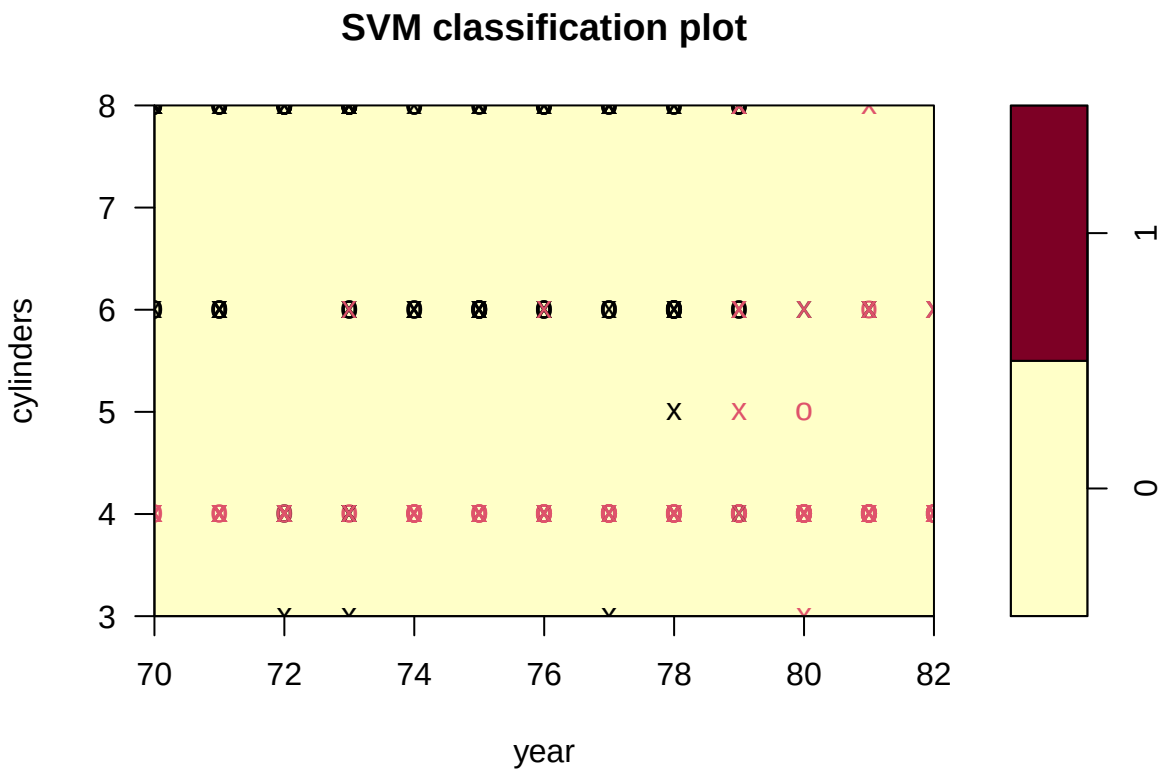

```
#radial
for(i in 1:2){
  for(k in (i+1):7){
    plot(svmfit_r, df,
         as.formula(paste0(names_list[i], "~", names_list[k])))
  }
}
```

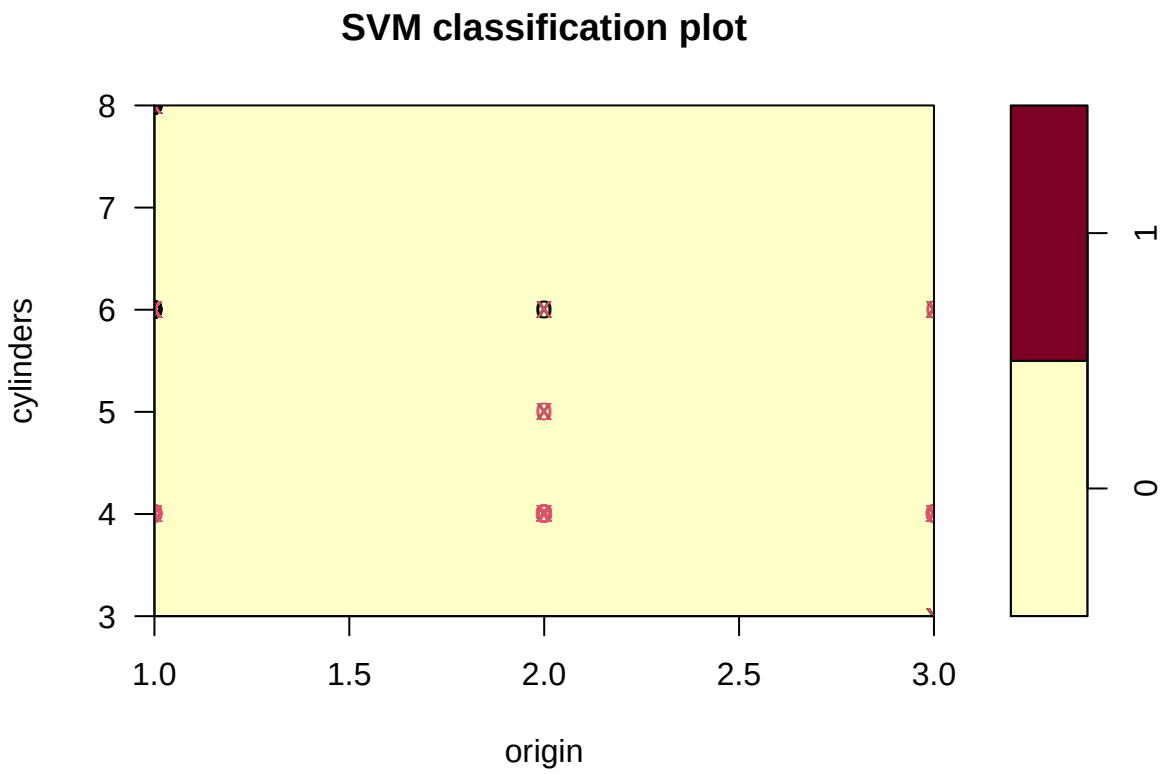

SVM classification plot
cylinders
displacement
8
7
6
5
4
3
100
150
200
250
300
350
400
450
1
0

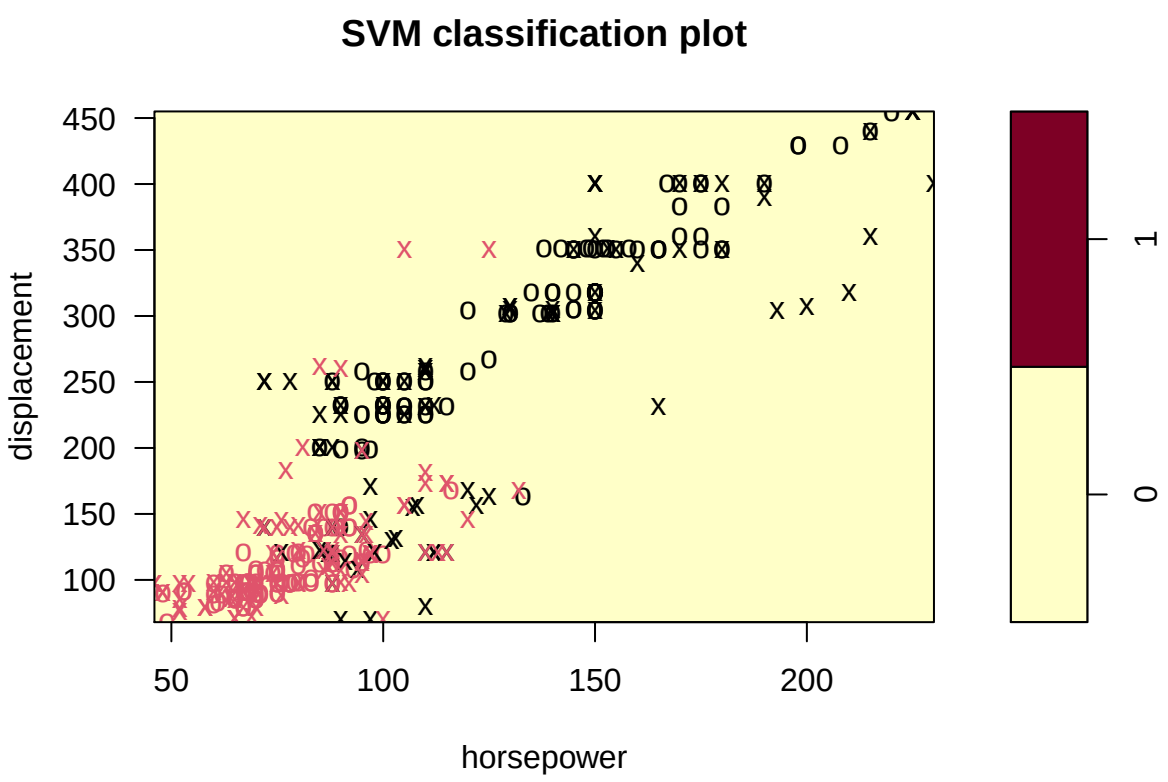

SVM classification plot
cylinders
horsepower
8
7
6
5
4
3
50
100
150
200
1
0

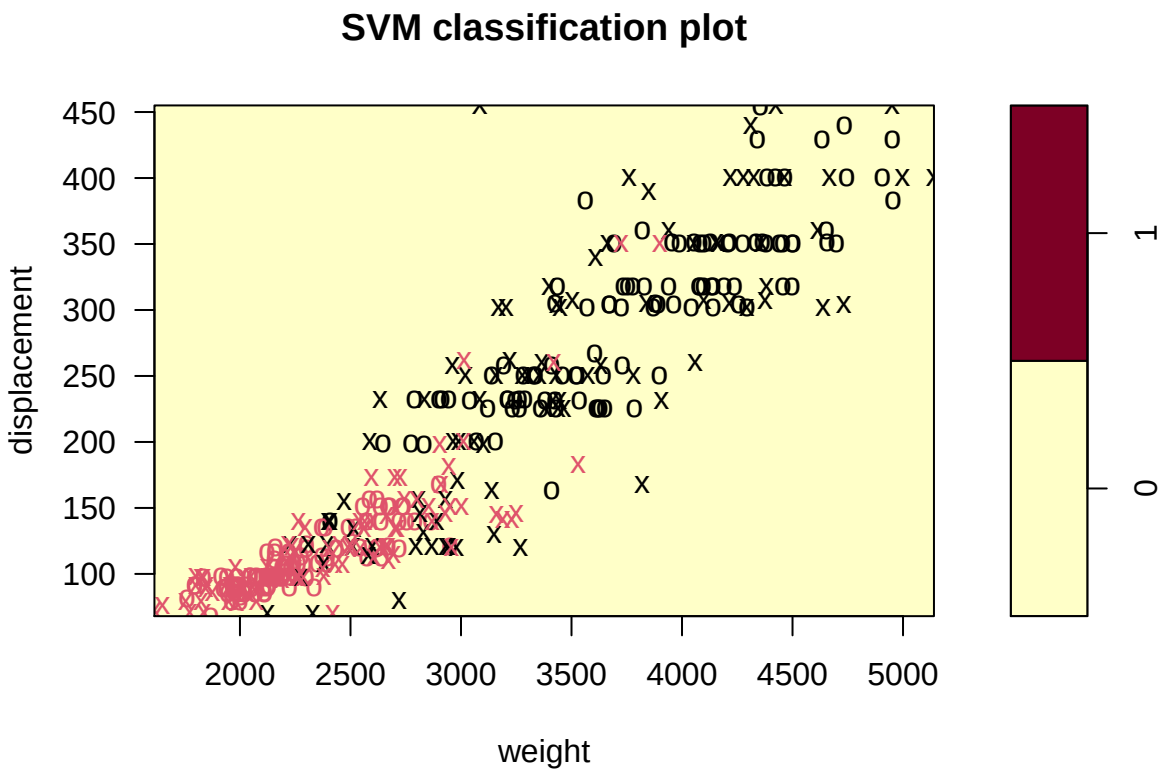

SVM classification plot
cylinders
weight
8
7
6
5
4
3
2000
2500
3000
3500
4000
4500
5000
1
0

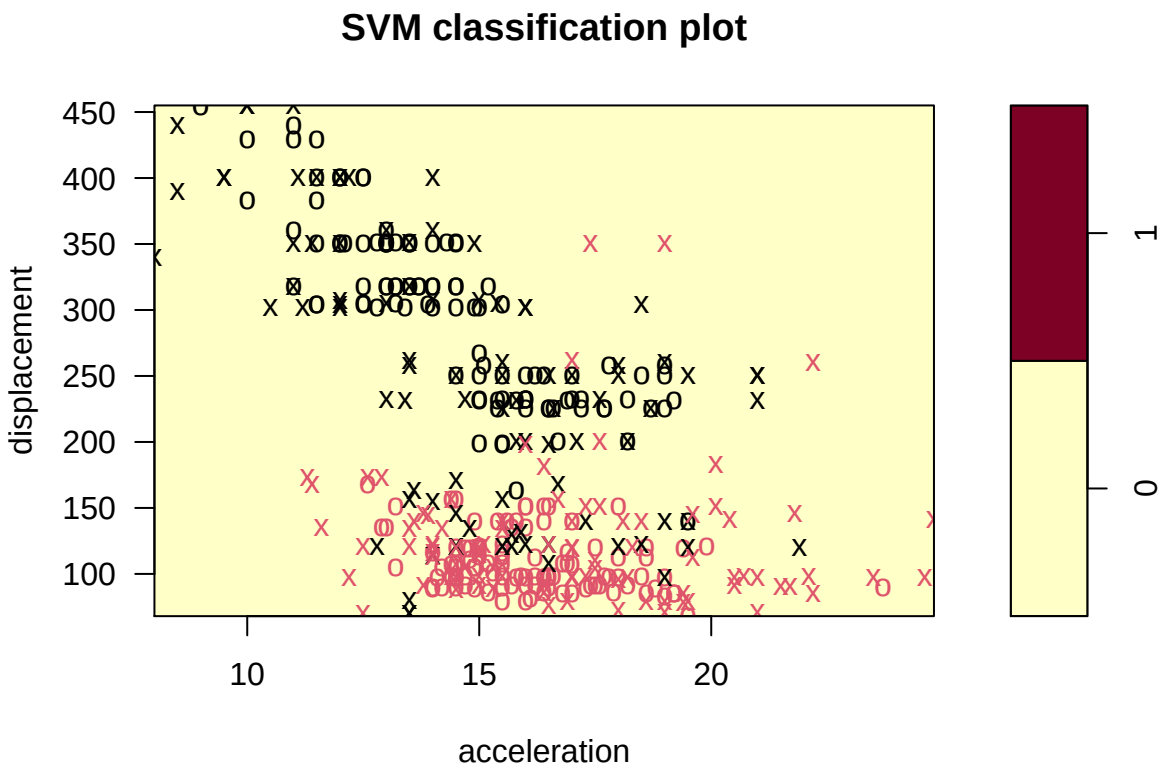

SVM classification plot
cylinders
acceleration
8
7
6
5
4
3
10
15
20
1
0

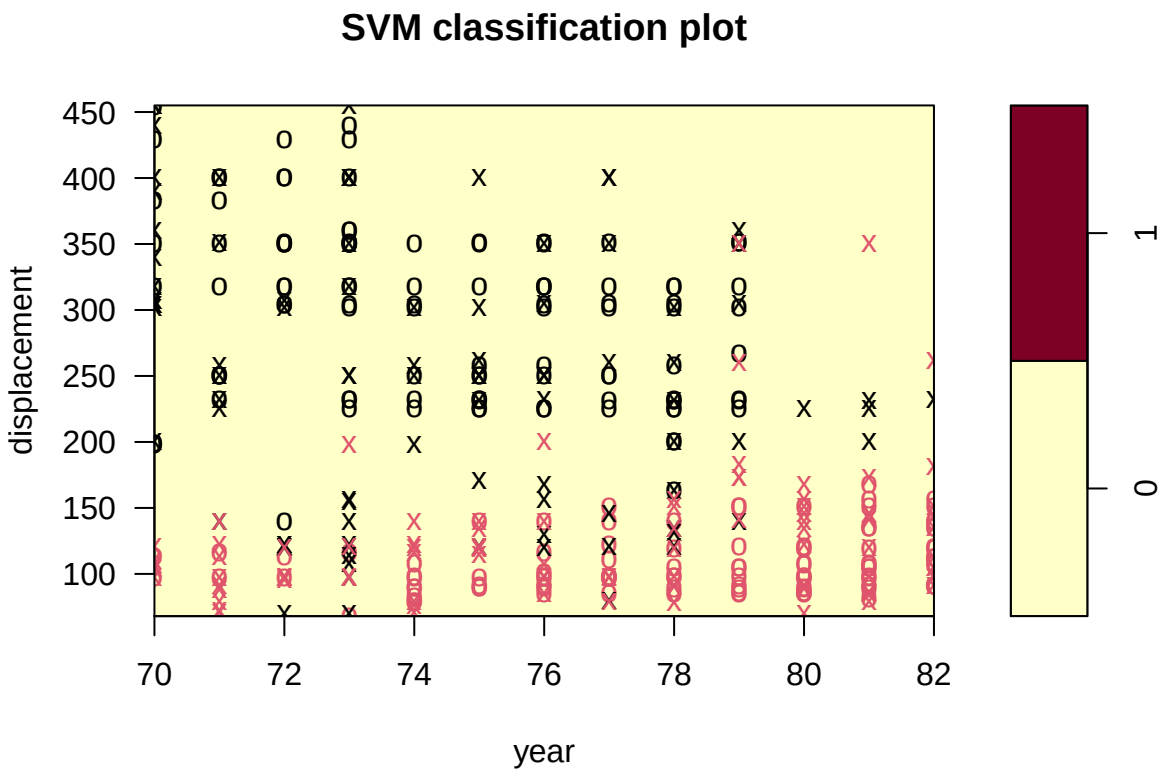

SVM classification plot
cylinders
year
8
7
6
5
4
3
70
72
74
76
78
80
82
1
0

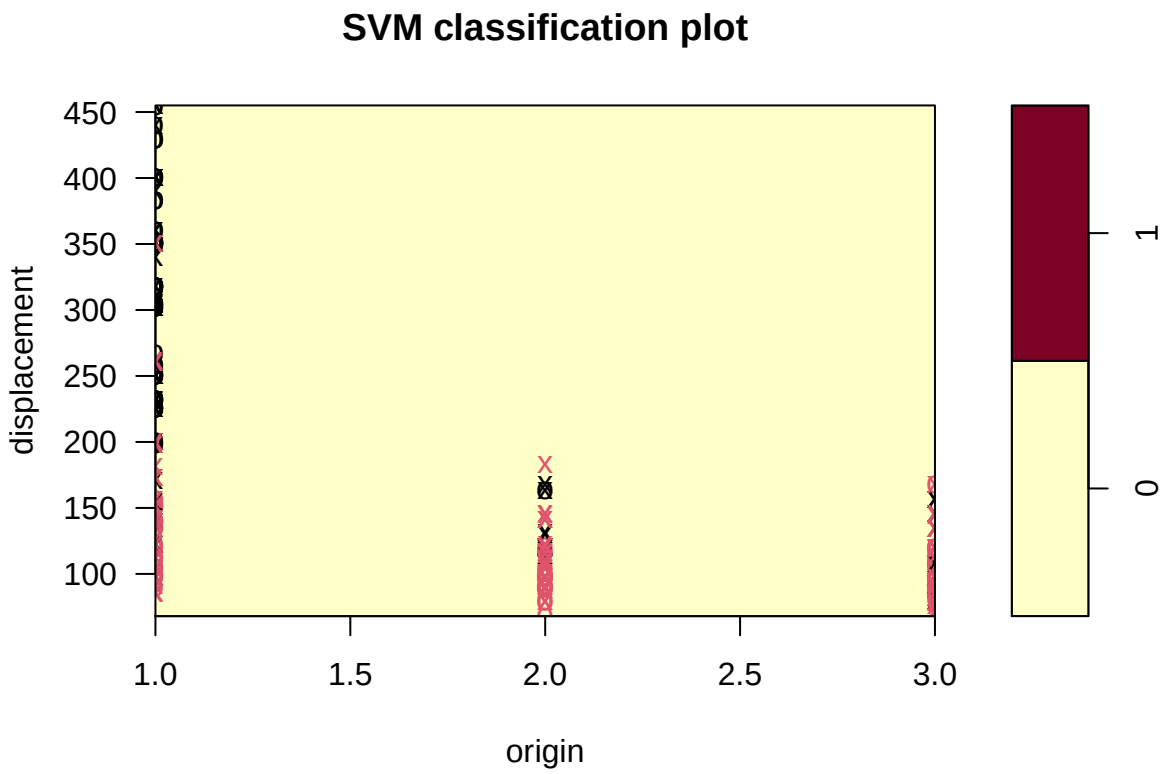

SVM classification plot
cylinders
origin
8
7
6
5
4
3
1.0
1.5
2.0
2.5
3.0
1
0


SVM classification plot
displacement
horsepower
450
400
350
300
250
200
150
100
50
100
150
200
1
0

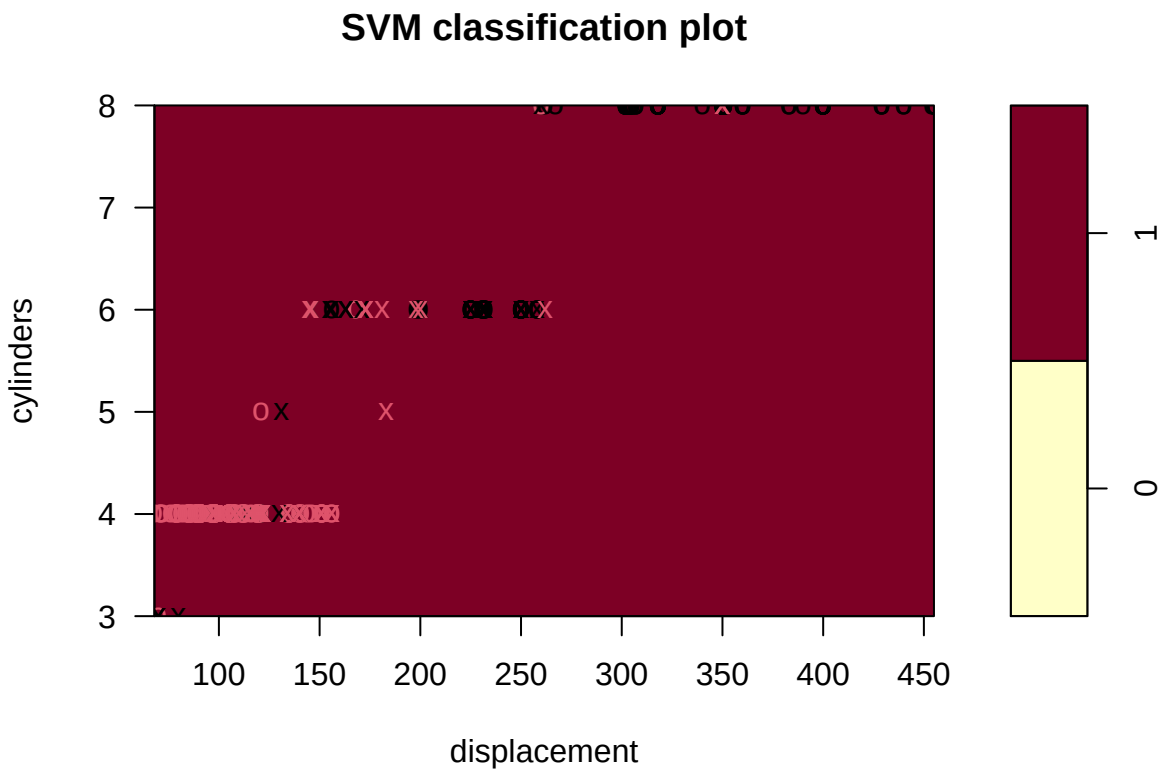

SVM classification plot
displacement
weight
450
400
350
300
250
200
150
100
2000
2500
3000
3500
4000
4500
5000
1
0

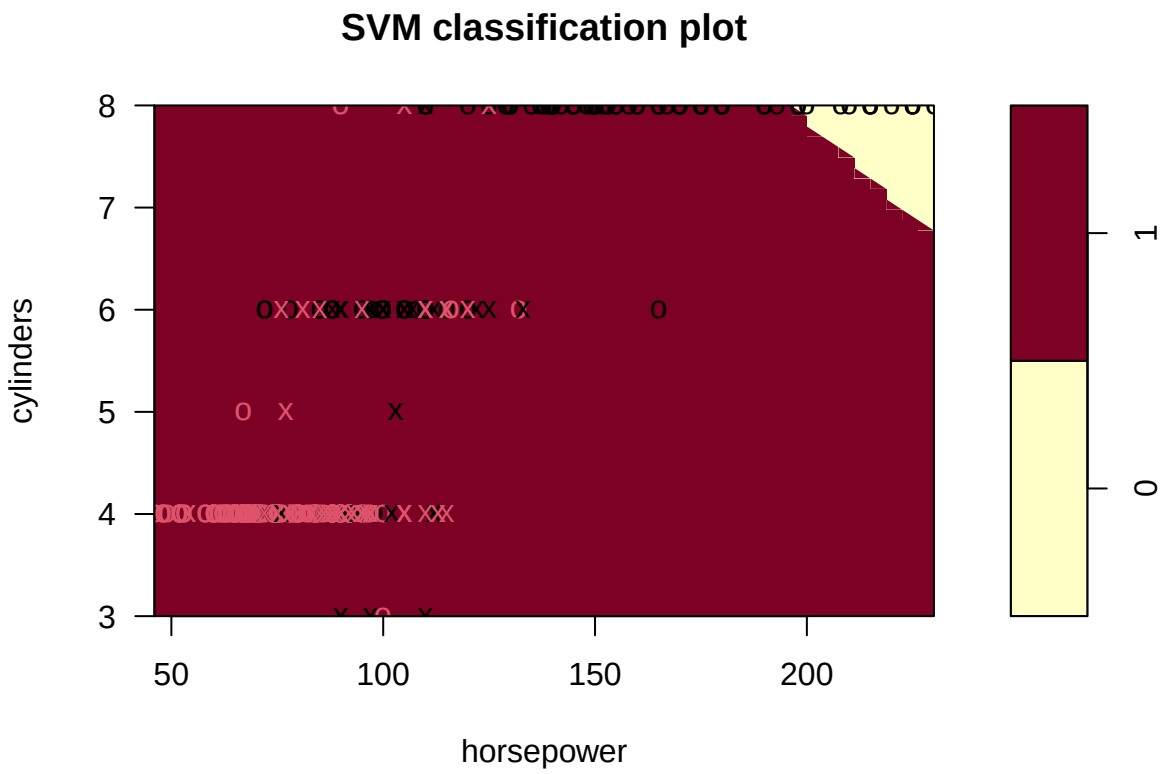

SVM classification plot
displacement
acceleration
450
400
350
300
250
200
150
100
10
15
20
1
0

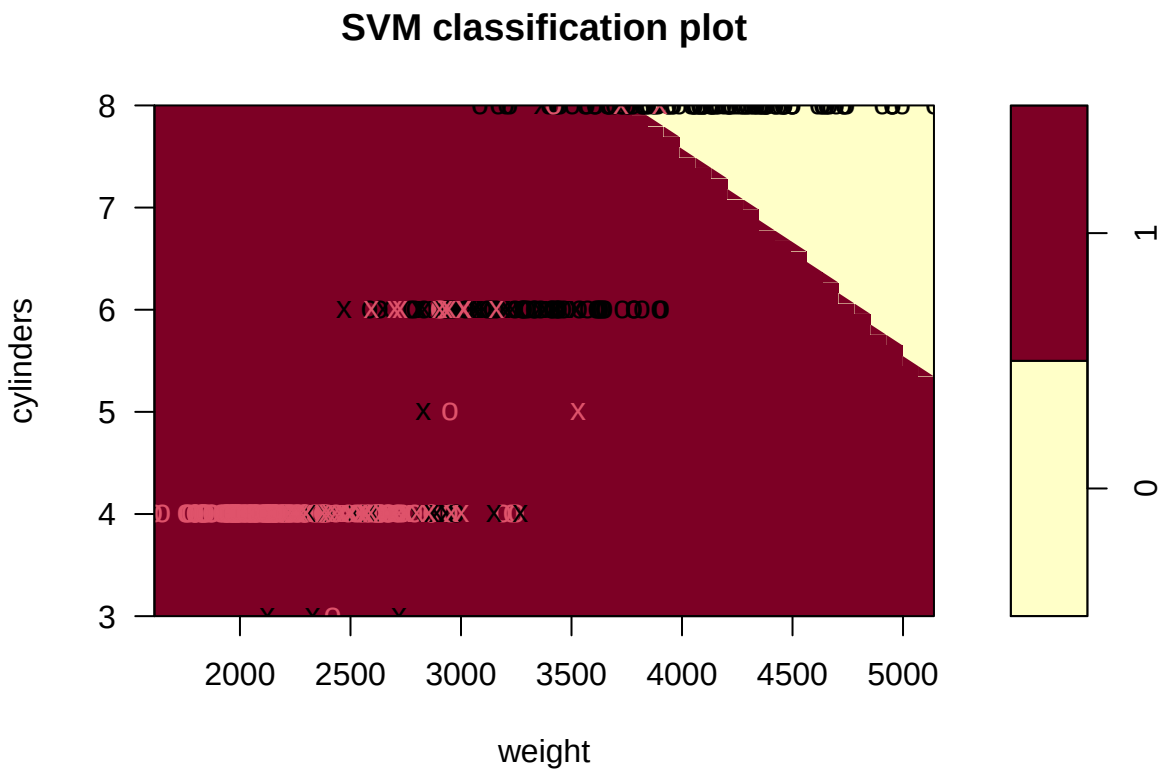

SVM classification plot
displacement
year
450
400
350
300
250
200
150
100
70
72
74
76
78
80
82
1
0

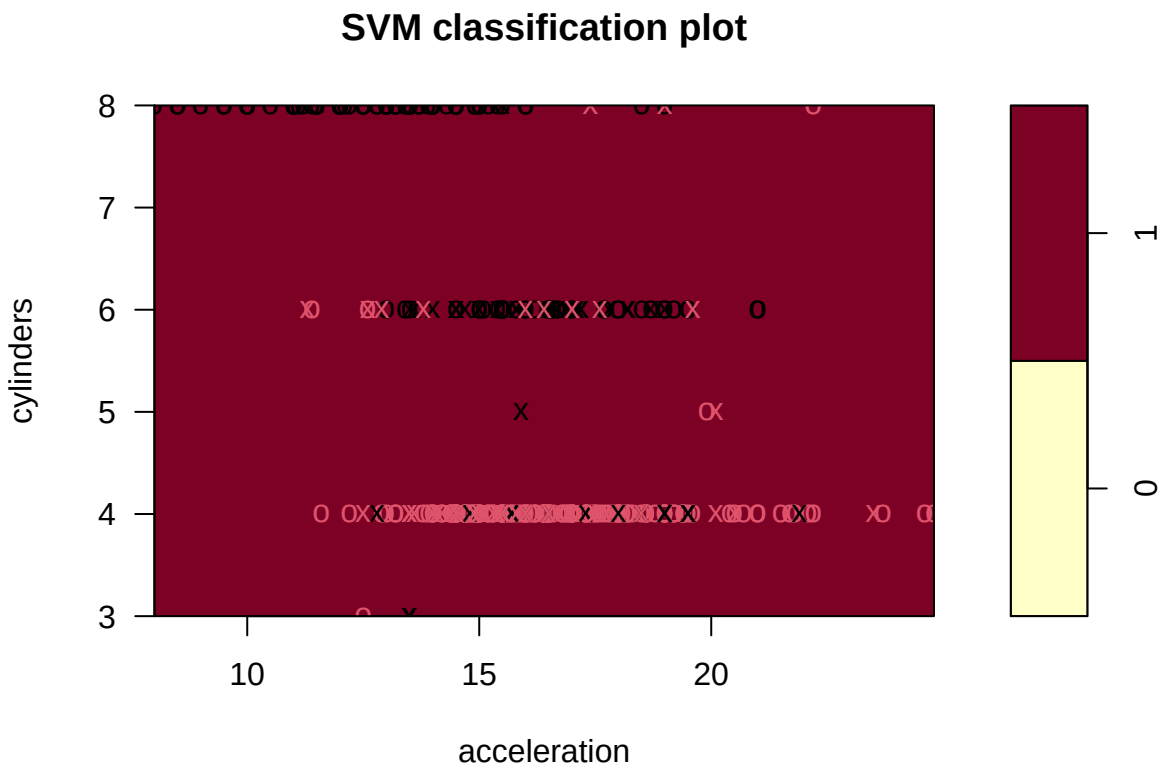


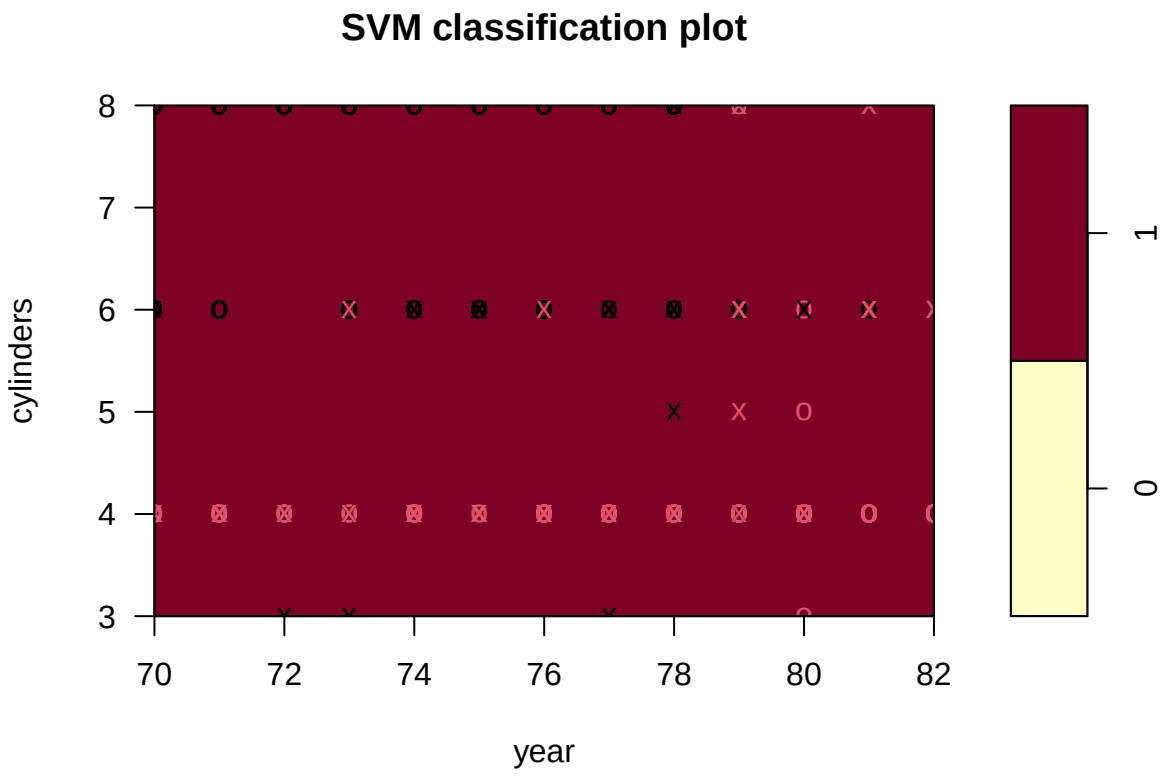

```
#polynomial
for(i in 1:2){
  for(k in (i+1):7){
    plot(svmfit_p, df,
         as.formula(paste0(names_list[i], "~", names_list[k])))
  }
}
```

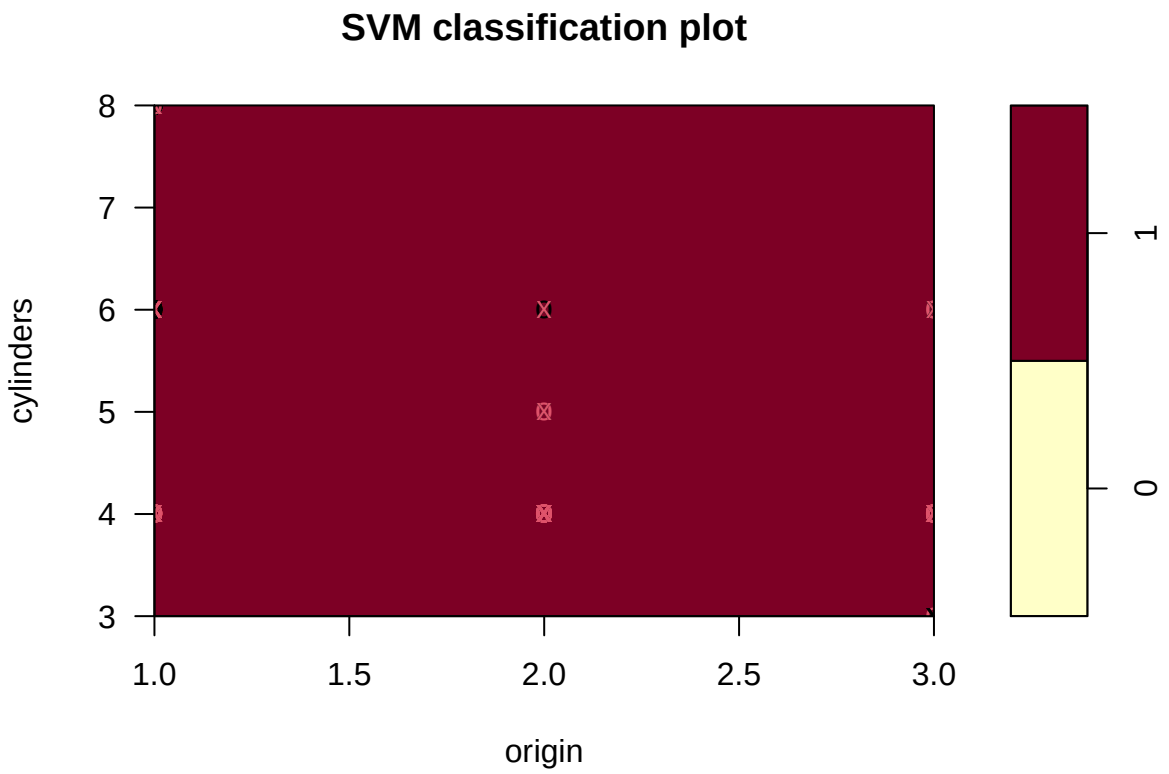

SVM classification plot
cylinders
displacement
8
7
6
5
4
3
100
150
200
250
300
350
400
450
1
0

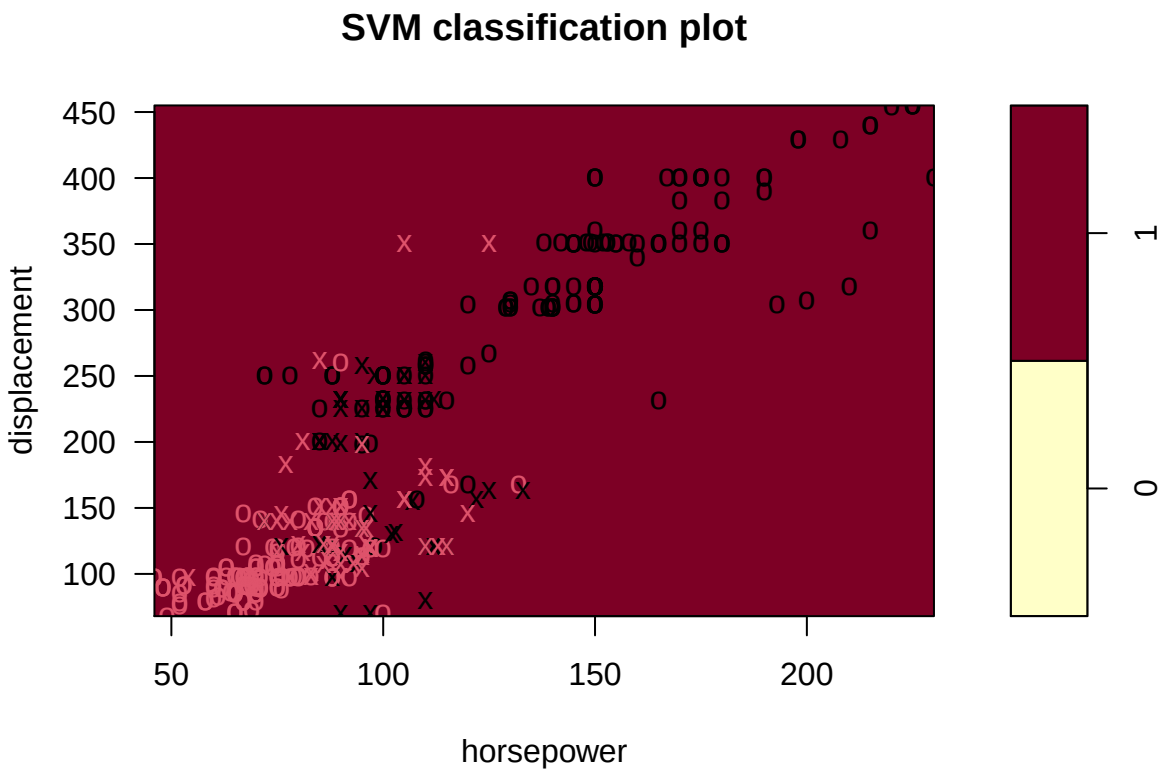

SVM classification plot
cylinders
horsepower
8
7
6
5
4
3
50
100
150
200
1
0

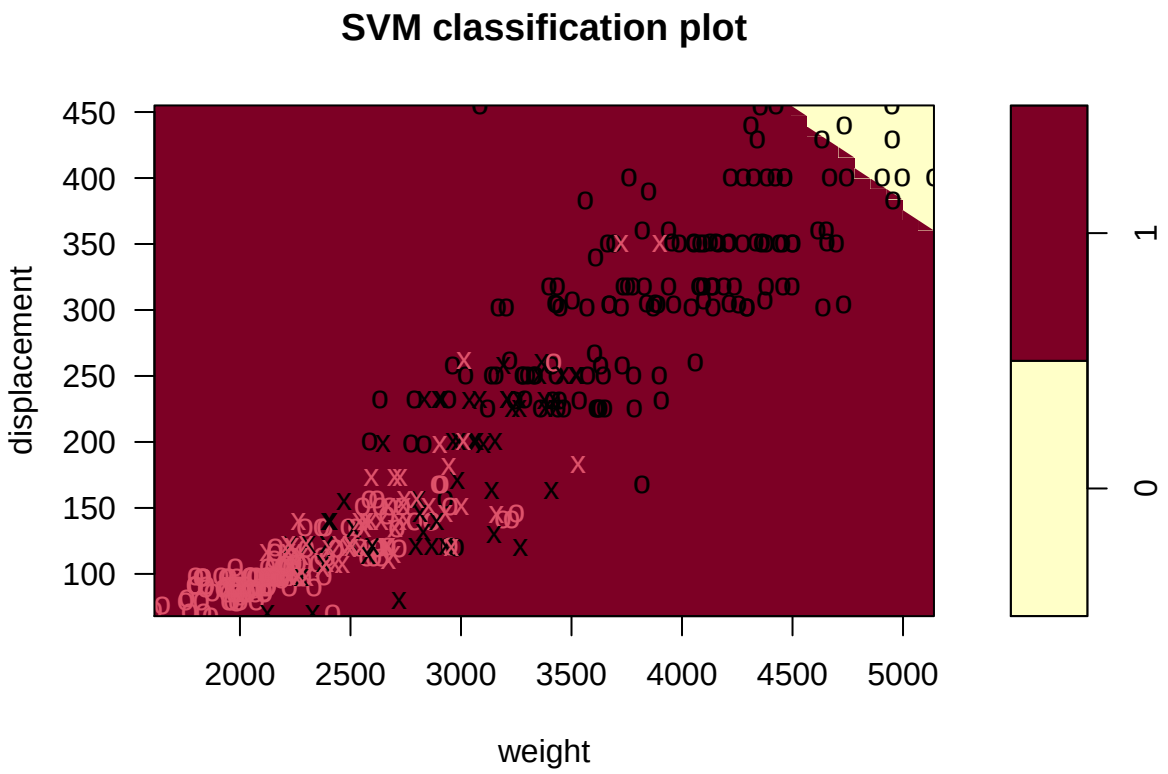

SVM classification plot
cylinders
weight
8
7
6
5
4
3
2000
2500
3000
3500
4000
4500
5000
1
0

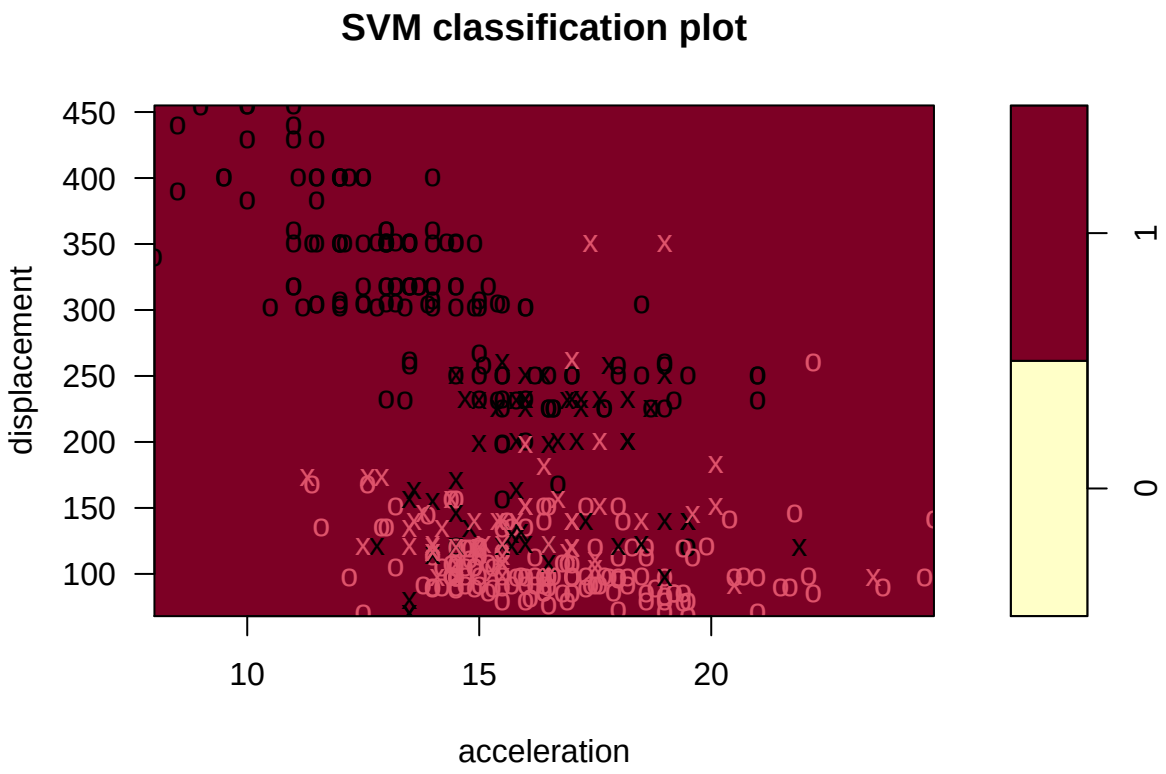

SVM classification plot
cylinders
acceleration
8
7
6
5
4
3
10
15
20
1
0

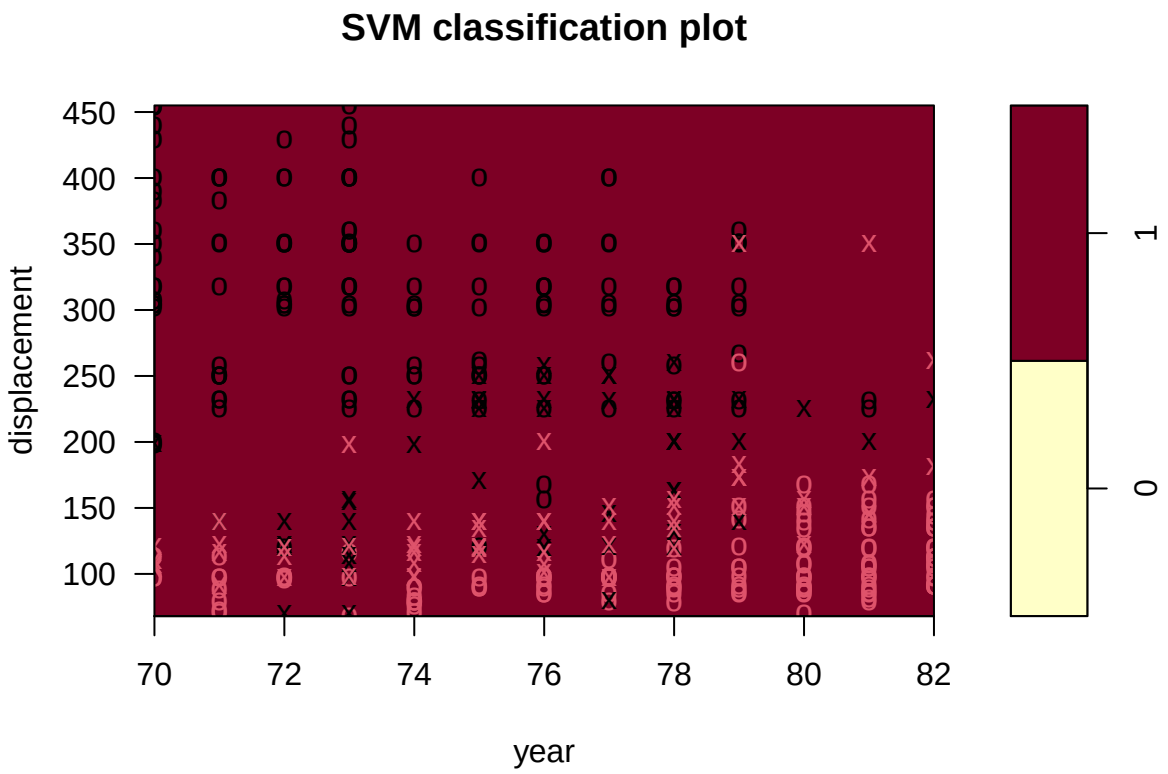

SVM classification plot
cylinders
year
8
7
6
5
4
3
70
72
74
76
78
80
82
1
0

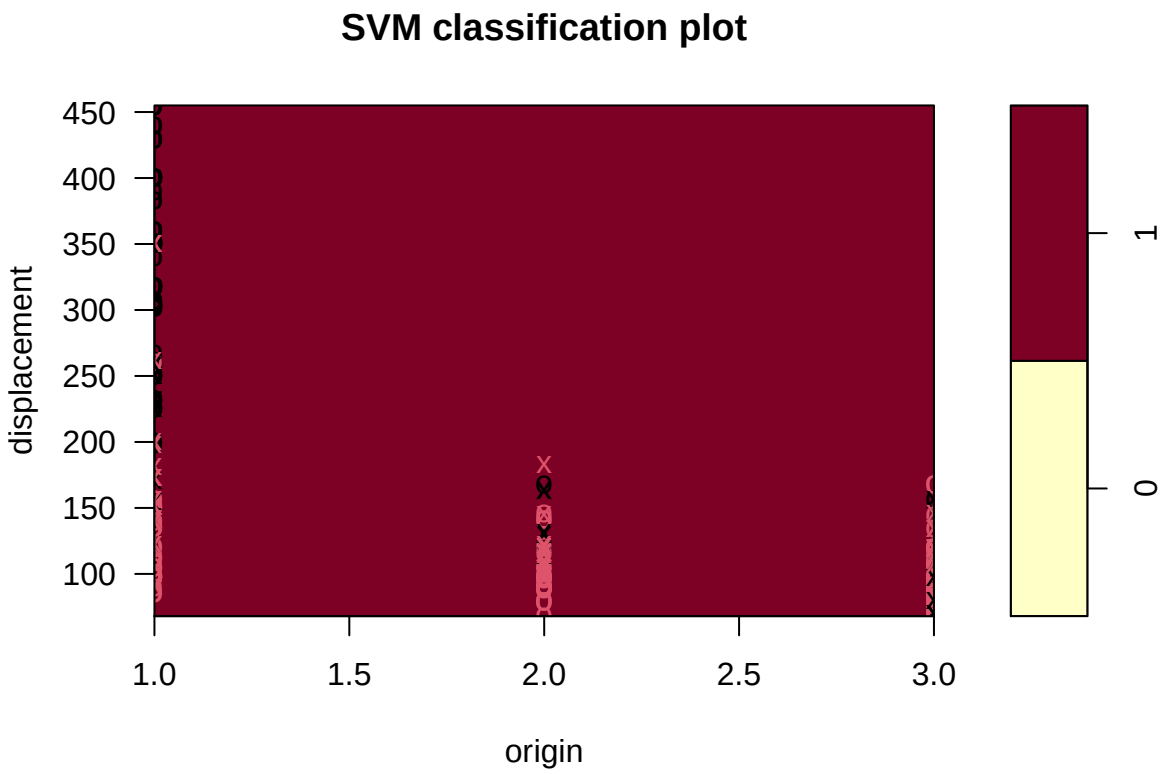

SVM classification plot
cylinders
origin
8
7
6
5
4
3
1.0
1.5
2.0
2.5
3.0
1
0


SVM classification plot
displacement
horsepower
450
400
350
300
250
200
150
100
50
100
150
200
1
0


SVM classification plot
displacement
weight
450
400
350
300
250
200
150
100
2000
2500
3000
3500
4000
4500
5000
1
0


SVM classification plot
displacement
450
400
350
300
250
200
150
100
10
15
20
acceleration
1
0


SVM classification plot
displacement
450
400
350
300
250
200
150
100
70
72
74
76
78
80
82
year
1
0


SVM classification plot
displacement
origin
450
400
350
300
250
200
150
100
1.0
1.5
2.0
2.5
3.0
1
0