STATISTICAL LEARNING

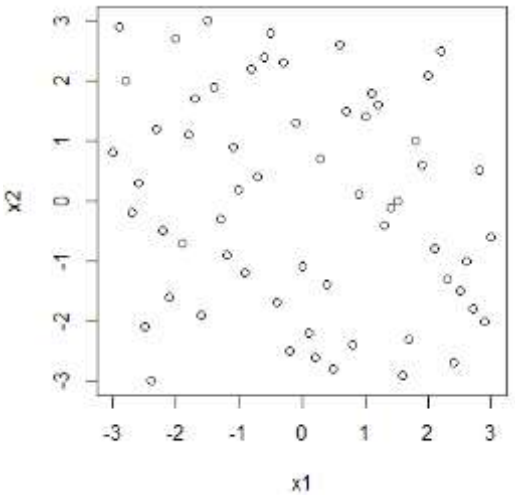
with application in R

Linear Regression
Classification (Logistic Regression, QDA)
Bootstrapping / Subset Selection/ Tree (random forest)
SVM (linear, radial kernel) / Clustering

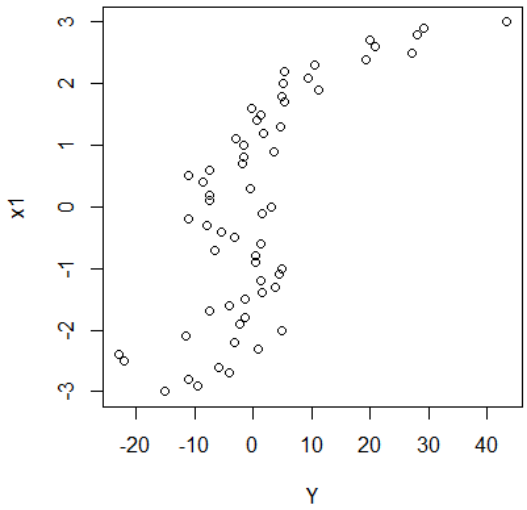
1. Preparation step

```
> library(MASS)
> set.seed(1)
> noise=rnorm(61,0,4)
> x1<-seq(-3,3,by=0.1)
> x2<-sample(x1,61,replace=FALSE)
> Y = -x1+x2-x2*x1+x1^2-x2^2+x1^3+noise
> Z<- as.factor(ifelse(Y >=0, 1, 0))
> df1<-data.frame(x1,x2,Y,Z)
```

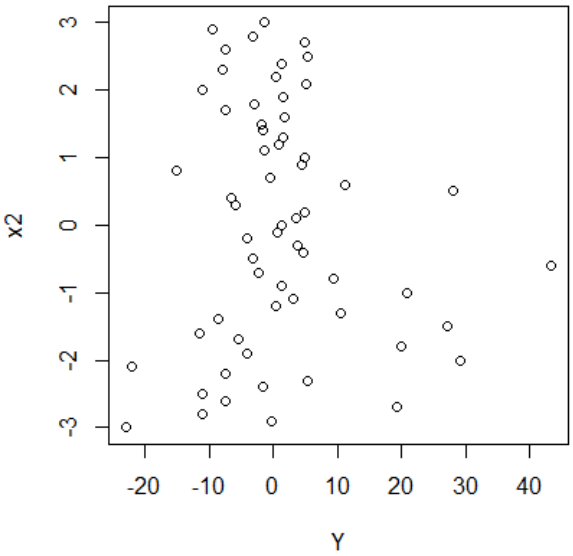
```
> plot(x1,x2)
```



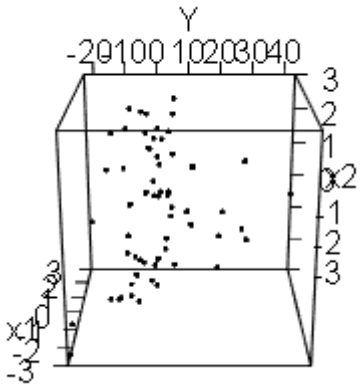

```
> plot(Y,x1)
```




```
> plot(Y,x2)
```




```
> library(rgl)
> plot3d(x1,x2,Y)
```



2. linear Regression

> lm.fit=lm(Y~x1,data=df1,subset=train)

> mean((Y-predict(lm.fit,df1))[-train]^2)

[1] 77.25359

> summary(lm.fit)

Residuals:

Min	1Q	Median	3Q	Max
-10.9025	-4.3669	-0.9712	3.5535	30.5958

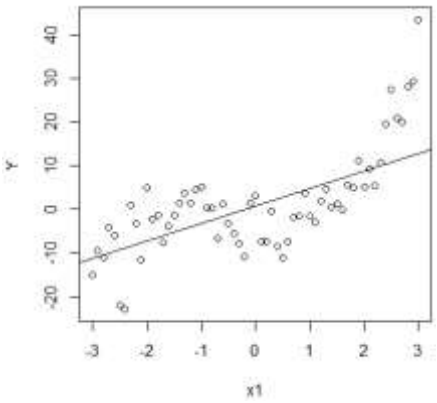
Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	0.7768	1.4818	0.524	0.604231	
x1	4.0246	0.9167	4.390	0.000146	***

Residual standard error: 8.052 on 28 degrees of freedom
Multiple R-squared: 0.4077, Adjusted R-squared: 0.3866
F-statistic: 19.28 on 1 and 28 DF, p-value: 0.0001464

> plot(x1,Y)

> abline(lm.fit)



> lm.fit2=lm(Y~x2,data=df1,subset=train)

> mean((Y-predict(lm.fit2,df1))[-train]^2)

[1] 172.3199

> summary(lm.fit2)

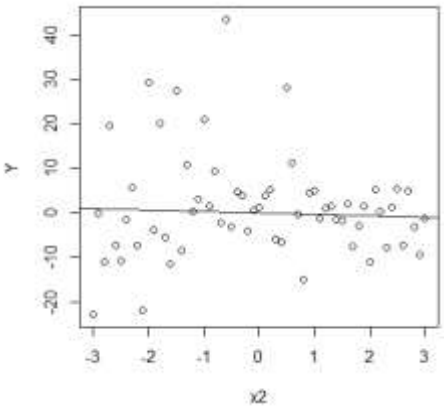
Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.05411	1.90712	-0.028	0.978
x2	-0.34371	1.08381	-0.317	0.753

Residual standard error: 10.44 on 28 degrees of freedom
Multiple R-squared: 0.003579, Adjusted R-squared: -0.03201
F-statistic: 0.1006 on 1 and 28 DF, p-value: 0.7535

> plot(x2,Y)

> abline(lm.fit2)



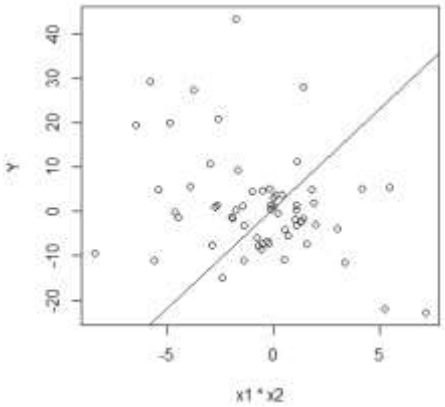
> lm.fit3=lm(Y~x1*x2,data=df1,subset=train)

> mean((Y-predict(lm.fit3,df1))[-train]^2)

[1] 67.75805

> plot(x1*x2,Y)

> abline(lm.fit3)



> summary(lm.fit3)

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	**0.5403**	**1.6356**	**0.330**	**0.743809**	
x1	**4.4921**	**0.9933**	**4.522**	**0.000119**	***
x2	**1.0578**	**0.9602**	**1.102**	**0.280721**	
x1:x2	**-0.3453**	**0.6187**	**-0.558**	**0.581513**	

Residual standard error: 8.021 on 26 degrees of freedom
Multiple R-squared: 0.4542, Adjusted R-squared: 0.3912
F-statistic: 7.213 on 3 and 26 DF, p-value: 0.00112

세개의 모델 중 MSE값은 lm.fit3이 제일 작다. 하지만 lm.fit3모델 중 x1* x2 변수의 유의확률 p값은 유의수준보

다 크므로 통계적으로 의미 있다고 볼 수 없다. X2변수도 마찬가지다. 모델의 설명력을 나타내는 r제곱 값도 다른 모델들에 비해서는 크지만 값이 상대적으로 크다고 보긴 힘들다.

3. Classification

Logistic regression

```
> library(ISLR)
> df1<-data.frame(x1,x2,Y,Z)
> df.test=df1[-train,]
> Z.test=df1[-train,]$Z
```

>glm.fit=glm(Z~x1+x2,data=df1, family=binomial, subset=train)

```
> summary(glm.fit)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-1.4722  -0.9144  -0.5573   0.8638   1.9889

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.36316    0.42549  -0.854   0.3934
x1           0.69355    0.30025   2.310   0.0209 *
x2          -0.02809    0.21850  -0.129   0.8977
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 40.381  on 29  degrees of freedom
Residual deviance: 33.505  on 27  degrees of freedom
AIC: 39.505
```

```
>glm.probs=predict(glm.fit,df.test,
type="response")
> glm.pred=rep("0",30)
> glm.pred[glm.probs>0.5]="1"
> table(glm.pred,Z.test)
         Z.test
glm.pred  0  1
       0  9  7
       1  3 12
```

> mean(glm.pred==Z.test)

[1] 0.6774194

QDA

qda.fit=qda(Z~x1+x2,data=df1,subset=train)

```
> qda.fit
Prior probabilities of groups:
  0   1
0.6 0.4

Group means:
           x1        x2
0 -0.8055556 0.1722222
1  0.7000000 0.2166667
2
> qda.test=predict(qda.fit,df.test)$class
> table(qda.test,Z.test)
         Z.test
qda.test  0  1
       0  9  7
       1  3 12
```

> mean(qda.test==Z.test)

[1] 0.6774194

4. Bootstrapping

```
> library(boot)
```

> median(Y)

[1] 0.04812321

```
> resample=matrix(sample(Y, size=61*20
 ,replace=TRUE), nrow=61)
> median=apply(resample,2,median)
> summary(median)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.57374 -0.61937  0.04812  0.12226  0.63365  3.68954
```

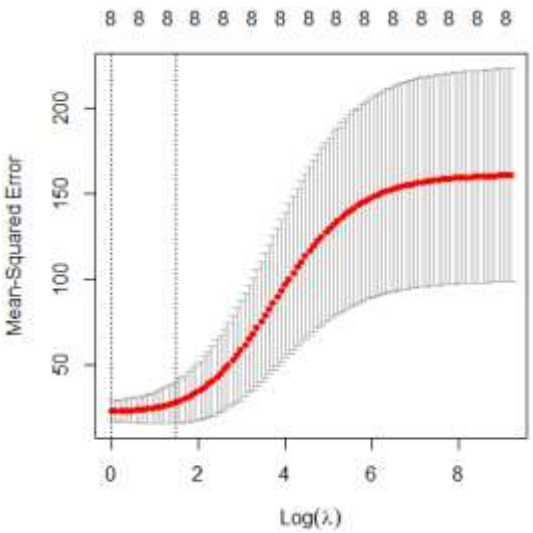
> quantile(median,c(0.025,0.975))

2.5% 97.5%

-1.57374 2.87810

5. Subset Selection

> df2<-data.frame
(Y,x1,x2,x1^2,x1^3,x2^2,x2^3,x1*x2,x1^2*x2^2)
> train=sample(61,30)
> df2.train=df2[train,]
> df2.test=df2[-train,]
> CV.X=model.matrix(Y~.,df2.train)
> CV.Y=df2.train$Y
> library(glmnet)
> ridge=cv.glmnet
(x=CV.X, y=CV.Y, family="gaussian", alpha=0, nfolds=5)
> plot(ridge)



> ridge$lambda.min
[1] 0.9960111
> log(ridge$lambda.min)
[1] -0.003996911
> ridge.gnet=glmnet
(x=CV.X, y=CV.Y, family="gaussian", alpha=0, lambda =ridge$lambda.min,nfolds=5)
> coef(ridge.gnet)

	s0
(Intercept)	0.35865094
(Intercept)	.
x1	0.41689260
x2	1.21907928
x1.2	1.96672121
x1.3	0.63556429
x2.2	-1.15820993
x2.3	-0.06186839
x1...x2	-0.73279393
x1.2...x2.2	-0.15744158

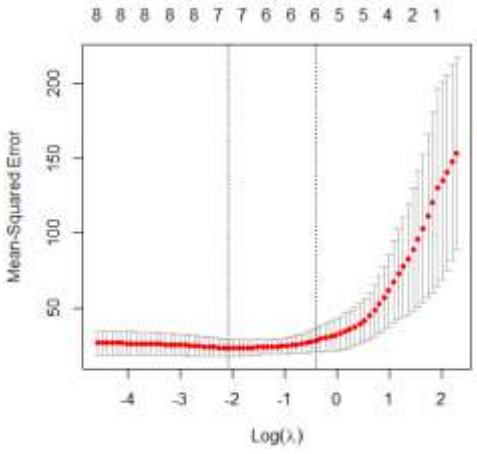
> CV.test.x=model.matrix(Y~.,df2.test)
> ridge.pred=predict(ridge.gnet,newx=CV.test.x)
> library(caret)
> postResample(pred=ridge.pred, obs=df2.test$Y)

RMSE	Rsquared	MAE
5.768105	**0.821921**	**4.466748**

> mean((ridge.pred-df2.test$Y)^2)
[1] 33.27103

LASSO

> set.seed(123)
>lasso=cv.glmnet(x=CV.X,y=CV.Y, family="gaussian" ,alpha=1,nfolds=5)
> plot(lasso)



> log(lasso$lambda.min)
[1] -2.073998
> log(lasso$lambda.1se)
[1] -0.3993903

1) lambda = lasso$lambda.min

```
lasso.gnet=glmnet(x=CV.X,y=CV.Y,family="gaus
sian",alpha=1,lambda = lasso$lambda.min)
> lasso.pred=predict(lasso.gnet,newx=CV.test.x)
> postResample(pred=lasso.pred,obs=df2.test$Y)
```

RMSE Rsquared MAE

5.3865436 0.8413887 4.3185550

```
> mean((lasso.pred-df2.test$Y)^2)
[1] 29.01485

> coef(lasso,lasso$lambda.min)
10 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept) -0.09835224
(Intercept)    .
x1             -0.35692957
x2              0.86668558
x1.2            1.99848129
x1.3            0.79909153
x2.2           -1.17014709
x2.3            .
x1...x2        -0.81942585
x1.2...x2.2 -0.12962756

>
```

2) lambda = lasso$lambda.1se

```
> coef(lasso,lasso$lambda.1se)
10 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)    0.93445852
(Intercept)    .
x1              .
x2              0.67415730
x1.2            1.42032387
x1.3            0.75759902
x2.2           -1.33365232
x2.3            .
x1...x2        -0.67733295
x1.2...x2.2 -0.01497805

lasso.gnet=glmnet(x=CV.X,y=CV.Y,family="gaus
sian",alpha=1,lambda = lasso$lambda.1se)
> lasso.pred=predict(lasso.gnet,newx=CV.test.x)
> postResample(pred=lasso.pred,obs=df2.test$Y)
```

RMSE Rsquared MAE

4.9794092 0.8721349 3.8977791

```
> mean((lasso.pred-df2.test$Y)^2)
[1] 24.79452
```

lambda.min, lambda.1se 중 후자의 에러가 더 낮고 R제곱 값은 더 크다. lambda 값으로는 lambda.1se 가 더 좋다.

Ridge 와 Lasso 모델을 비교하면 lasso 모델이 전체적으로 RMSE가 더 낮다. Ridge는 비효율적인 변수가 있더라도 제거를 하지 않는다. 반면 lasso는 영향력이 없는 변수는 제거를 하기에 더 효율적이라 볼 수 있다.

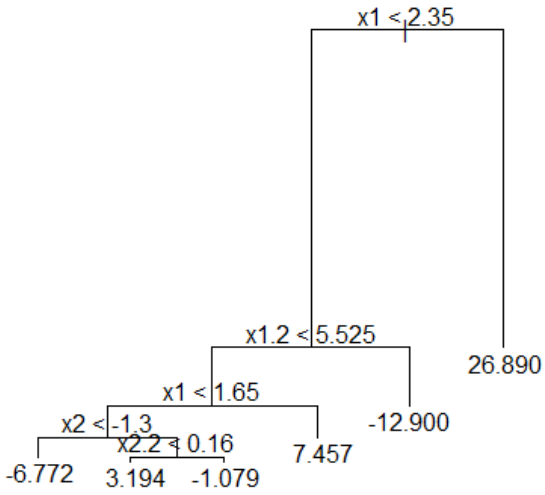
6. Tree

Regression Tree model

```
> library(tree)
> cv.tree =cv.tree(tree)
$size
[1] 6 5 4 3 2 1

$dev
[1] 3169.264 3213.439 3669.348 3990.859
[5] 4132.779 9035.333
$k
[1]         -Inf    87.62031    321.57455
[4]   517.28787   940.32057 5201.07972
$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

terminal node가 6개일때 에러가 제일 작다.

```
> prune.tree=prune.tree(tree,best=6)
> plot(prune.tree)
> text(tree,pretty=0)
```



Train / Test sample 로 나누기

```
> train=sample(61,30)
> df2.train=df2[train,]
> df2.test=df2[-train,]
> tree=tree(Y~.,df2,subset=train)
> cv.tree=cv.tree(tree)
> cv.tree
$size
[1] 4 2 1
$dev
[1] 2936.002 2961.118 3590.941
$k
[1]       -Inf   109.5927 1320.5066
$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"

> prune.tree.pred=
predict(prune.tree,newdata=df2.test)
> mean((prune.tree.pred-df2.test$Y)^2)
[1] 59.12797
```

Random Forest

```
> library(randomForest)
> set.seed(2)
> tree.random= randomForest
(Y~.,data=df2,subset=train,mtry=3,importance=TRUE)
> tree.random

Type of random forest: regression
                      Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 86.47251
                    % Var explained: 15.36

> tree.pred=
predict(tree.random, newdata=df2.test)
> plot(tree.pred,df2.test$Y)
> abline(0,1)
> mean((tree.pred-df2.test$Y)^2)
[1] 50.36067
```

RandomForest로 진행한 모델의 RMSE가 더 낮다.

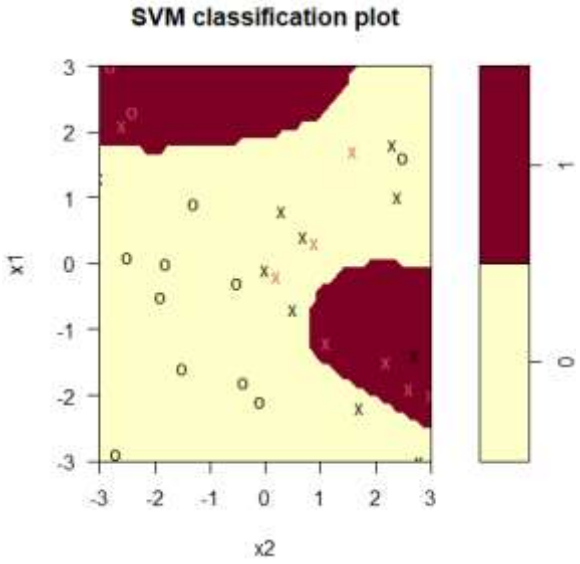
7. SVM

LINEAR

```
> library(e1071)
> set.seed(10)
> noise=rnorm(61,0,4)
> x1<-seq(-3,3,by=0.1)
> x2<-sample(x1,61,replace=FALSE)
> Y = -x1+x2-x2*x1+x1^2-x2^2+x1^3+noise
> Z<- as.factor(ifelse(Y >=0, 1, 0))
> df=data.frame(x1,x2,Z)
> df.train=df[train,]
> df.test=df[-train,]
```

> svmfit=
svm(Z~.,data=df.train,kernal="linear",cost=10)

```
> plot(svmfit,df.train)
```




```
> summary(svmfit)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  18
 ( 8 10 )
Number of Classes:  2
Levels:
 0 1
> svmfit.pred=predict(svmfit,newdata=df2.test)
> svmfit.pred
 3  4  5  6  7  8 14 18 20 21 22 23 25 27 33
36 37 38 42 43
 0  0  0  0  0  0  1  0  1  0  0  1  1  0
0  0  0  0  0  0
45 46 50 51 53 55 56 57 58 59 60
 0  0  0  0  0  1  1  1  1  1  1
Levels: 0 1
> table(true=df.test$Z, pred=svmfit.pred)
      pred
true  0  1
   0 16  3
   1  5  7
```

TEST ERROR = (3+5)/(16+3+5+7)=25%

RADIAL KERNAL

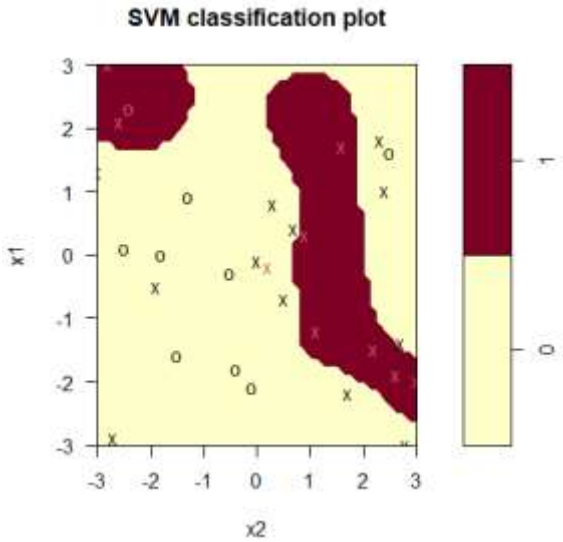
```
> set.seed(1)
>
```

svmfit=tune(svm,Z~.,data=df.train,kernal="radial",ranges=list(cost=c(0.1,1,10,20),gamma=c(0.5,1,2,3)))

```
> summary(svmfit)   #best : cost 20, gamma 0.5

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10     2
- Detailed performance results:
   cost gamma       error dispersion
1   0.1   0.5 0.3333333  0.3513642
2   1.0   0.5 0.4000000  0.2629369
3  10.0   0.5 0.3333333  0.3142697
4  20.0   0.5 0.3666667  0.2459549
5   0.1   1.0 0.3333333  0.3513642
```

6 1.0 1.0 0.3666667 0.2459549
7 10.0 1.0 0.3666667 0.2459549
8 20.0 1.0 0.4000000 0.3063122
9 0.1 2.0 0.3333333 0.3513642
10 1.0 2.0 0.3666667 0.3314763
11 10.0 2.0 0.3000000 0.2918650
12 20.0 2.0 0.3333333 0.3513642
13 0.1 3.0 0.3333333 0.3513642
14 1.0 3.0 0.3333333 0.2721655
15 10.0 3.0 0.3333333 0.3513642
16 20.0 3.0 0.3666667 0.3990730
- best performance: 0.3

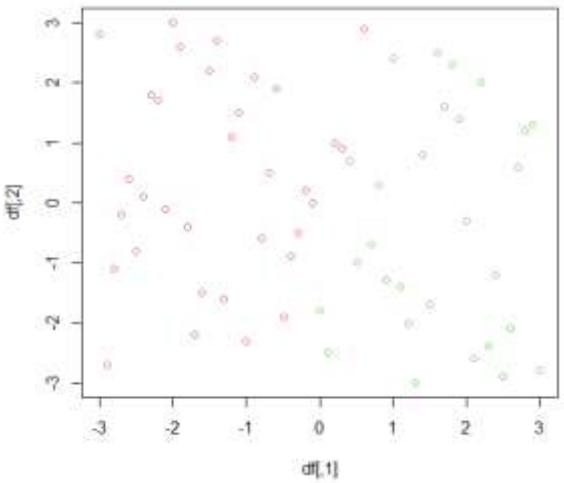
> **svmfit=**
svm(Z~.,data=df.train,kernal="radial",cost=10,gamma=2)
> plot(svmfit,df.train)



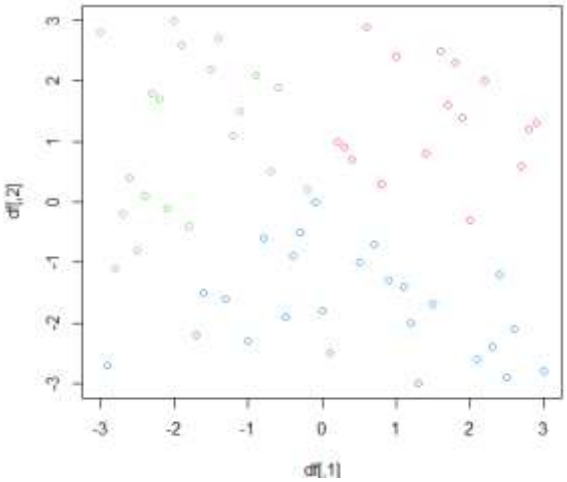
> km.out=kmeans(df,3,nstart=20)
> plot(df,col=(km.out$cluster+1))
> svmfit.pred=predict(svmfit,newdata=df2.test)
> table(true=df.test$Z, pred=svmfit.pred)
pred
true 0 1
0 14 5
1 5 7

8. Clustering

> set.seed(1)
> v1=matrix(x1,nrow = 61,ncol=1)
> v2=matrix(x2,nrow = 61,ncol=1)
> df=cbind(v1,v2)
> km.out=kmeans(df,2,nstart=20)
> plot(df,col=(km.out$cluster+1))
> km.out$tot.withinss
[1] 231.6468



> km.out=kmeans(df,3,nstart=20)
> plot(df,col=(km.out$cluster+1))
> km.out$tot.withinss
[1] 142.0731



K=3일때의 total within-cluster sum of squares가 더 작다.

> hc.complete=
hclust(dist(df),method="complete")
> plot(hc.complete)