```
#=========================
# polynomials
#=========================

load("poly.Rdata")
plot( y ~ x , train)
fit <- lm( y ~ poly(x, degree=2), train)
yhat <- predict(fit, newdata=test)
lines( yhat ~ x, train)
MSE.tr <- mean( (train$y - yhat)^2 )
n <- nrow(train)
ds = 0:20
fun <- function(d) if (d==0) lm(y~1, train) else
lm(y~poly(x,degree=d), train)
fits <- sapply(ds, fun)
MSEs.tr <- unlist( lapply(fits, deviance) )/n
plot(ds, MSEs.tr, type="b")
yhats <- lapply(fits, predict)
MSEs.te <- unlist( lapply(yhats,
           function(yhat) mean((test$y - yhat)^2)) )
plot(ds, MSEs.tr, type="b", col=4)
lines(ds, MSEs.te, type="b", col=2)
fit <- lm( y ~ poly(x, degree=14), train)
yhat <- predict(fit, newdata=test)
plot( y ~ x , train, col=4)
lines(yhat ~ x, train)
plot( y ~ x , test, col=2)
lines(yhat ~ x, test)
plot(y~x, truef, type="l", col=3)
points(y~x,train, col=4)
points(y~x,test, col=2)

AICs <- unlist( lapply(fits, AIC) )
BICs <- unlist( lapply(fits, BIC) )
plot(ds, AICs, type="b", col=5)
lines(ds, BICs, type="b", col=6)

d = 5
K<-5
set.seed(123)
folds <- sample( rep(1:K,length=n) )
KfoldCV <- vector()
for (k in 1:K){
  fit <- lm(y~poly(x,degree=d), train, subset=which(folds!=k))
  x.out <- train$x[which(folds==k)]
  yhat <- predict(fit, newdata=list(x=x.out))
  y.out <- train$y[which(folds==k)]
  KfoldCV[k]<- mean( ( y.out - yhat )^2 )
}
mean(KfoldCV)
library(boot)
set.seed(123)
KCV = sapply(1:10, function(d)
     cv.glm(train, glm(y~poly(x,degree=d),
```

```
     train, family = gaussian), K=5 )$delta[1] )
plot(1:10, KCV, type="b", log="y")
d = 5
oneout <- vector()
for (i in 1:n){
fit_i <- lm( y~poly(x,degree=d), data=train[-i,])
yhat_i <- predict(fit_i, newdata=data.frame(x=train$x[i]) )
oneout[i] <- ( train$y[i] -  yhat_i )^2
}
mean(oneout)
deg = 5
fit <-  lm(y~poly(x,deg), train)
X <- model.matrix(fit)
H <- X %*% solve(t(X)%*% X) %*% t(X)
LOOCV = mean(
   ( (train$y - predict(fit)) / (1-diag(H))  )^2
)
LOOCV
library(boot)
LOOCV = sapply(1:10, function(d)
     cv.glm(train, glm(y~poly(x,degree=d),
     train, family = gaussian) )$delta[1] )
GCV = MSEs.tr/(1-(ds+1)/n )^2
plot(1:10, LOOCV, type="b", log="y", col=5)
lines(ds, GCV, col=6)

#==========================
# auto
#==========================

auto <- read.table("auto.dat", header=TRUE, quote="\"")
plot(city.distance ~ engine.size, auto)
library(FNN)
k <- 10
x0 = seq(min(auto$engine.size),max(auto$engine.size), length.out =
200)
fit.kNN <- knn.reg(train=auto$engine.size, y=auto$city.distance,
test=data.frame(city.distance=x0),  k=k)
y0.kNN <- fit.kNN$pred
plot(city.distance ~ engine.size, auto)
lines(y0.kNN ~ x0, type="s", col=4)
library(sm)
h <- 0.5
fit.lr <- sm.regression(x=auto$engine.size, y=auto$city.distance,
h=h, eval.points=x0, display="se" )
s = 0.75
fit.loess = loess(city.distance ~ engine.size, auto, degree=1,
span=s)
y0 = predict(fit.loess, newdata=data.frame(engine.size=x0), se=TRUE)
plot(city.distance ~ engine.size, auto)
lines(y0$fit ~ x0, col=4)
lines(x0, y0$fit + 2*y0$se.fit , col=4, lty=3)
lines(x0, y0$fit - 2*y0$se.fit , col=4, lty=3)
library(splines)
```

```
knots = c(1.5,2,2.5)
d = 1
fit.splines = lm(city.distance ~ bs(engine.size, knots=knots,
degree=d), auto)
y0 = predict(fit.splines, newdata=data.frame(engine.size=x0),
se=TRUE)
plot(city.distance~engine.size, auto)
lines(x0, y0$fit, col=4)
lines(x0, y0$fit +2* y0$se ,lty =2, col=4)
lines(x0, y0$fit -2* y0$se ,lty =2, col=4)
fit.ns = lm(city.distance ~ ns(engine.size, knots=knots), auto)
y0 = predict(fit.ns, newdata=data.frame(engine.size=x0), se=T)
plot(city.distance~engine.size, auto)
lines(x0, y0$fit, col=4)
lines(x0, y0$fit +2* y0$se ,lty =2, col=4)
lines(x0, y0$fit -2* y0$se ,lty =2, col=4)
library(ggplot2)
ggplot(data = auto) + geom_point(mapping = aes(x = engine.size, y =
city.distance))
ggplot(data = auto) + geom_point(mapping = aes(x = engine.size, y =
city.distance), position = "jitter")
ggplot(data = auto, mapping = aes(x = engine.size, y =
city.distance, color = fuel)) +  geom_point()+
geom_smooth(method="loess", span = 0.75)
ggplot(data = auto) + geom_point(mapping = aes(x = engine.size, y =
city.distance)) + facet_wrap(  ~ fuel)


#=========================
# titanic
#=========================

train <- read.csv("titanic_tr.csv",
         header = TRUE, stringsAsFactors = FALSE)
test <- read.csv("titanic_te.csv",
         header = TRUE, stringsAsFactors = FALSE)
combi <- rbind(train, test)
str(combi)
combi$pclass <- as.factor(combi$pclass)
combi$sex <- as.factor(combi$sex)
combi$embarked <- as.factor(combi$embarked)
combi$survived01 <- combi$survived
combi$survived <- as.factor(combi$survived01)
levels(combi$survived) = c("Death","Alive")
combi$cabin[combi$cabin==""] <- NA
summary(combi)
combi[which(is.na(combi$embarked)), ]
boxplot(fare~ pclass + embarked, data=combi); abline(h=80)
combi$embarked[which(is.na(combi$embarked))] <- c("C","C")
combi[which(is.na(combi$fare)), ]
aggregate(fare ~ pclass + embarked, combi, FUN=median)
combi$fare[which(is.na(combi$fare))] <- 8.0500
aggregate(age ~ pclass + sex, combi, FUN=mean)
fit.age <- lm(age ~ sex + pclass,
```

```
          data = combi[!is.na(combi$age),])
combi$age[is.na(combi$age)] <- predict(fit.age,
          newdata=combi[is.na(combi$age),])
n = nrow(train)
m = nrow(test)
train <- combi[1:n,]
test <- combi[(n+1):(n+m),]
plot(survived~pclass, train)
plot(survived ~ age, train)
library(ggplot2)
ggplot(train, aes(x=age, y=survived01)) + geom_jitter(height = 0.05)
+ geom_smooth()
library(rpart)
fit.rpart <- rpart(survived ~ pclass + age, train,
               control=rpart.control(maxdepth =  3))
library(rpart.plot)
rpart.plot(fit.rpart, type=0, extra=2)
yhat <- predict(fit.rpart, newdata=test, type="class")
table(yhat, test$survived)
mean(yhat == test$survived)
library(ggplot2)
ggplot(train, aes(x = sex, fill = survived)) + geom_bar() +
facet_wrap(~ pclass)
ggplot(train, aes(x = age, fill = survived)) + facet_wrap(~sex +
pclass) + geom_histogram(binwidth = 10)
combi$name[1]
library(dplyr)
library(stringr)
combi <- combi %>%
     mutate(title = str_extract(name, "[a-zA-Z]+\\."))
table(combi$title)
titles <- data.frame(title = c("Mr.", "Capt.", "Col.", "Don.",
"Dr.", "Jonkheer.", "Major.", "Rev.", "Sir.",
                               "Mrs.", "Dona.", "Lady.", "Mme.",
"Countess.",
                               "Miss.", "Mlle.", "Ms.",
                               "Master."),
                     title2 = c(rep("Mr.", 9),
                                rep("Mrs.", 5),
                                rep("Miss.", 3),
                                "Master."),
                            stringsAsFactors = FALSE)
titles
combi <- combi %>%
  left_join(titles, by = "title")
combi %>%
  filter((sex == "female" & (title2 == "Mr." | title2 == "Master."))
|
           (sex == "male" & (title2 == "Mrs." | title2 == "Miss.")))
which(combi$name=="Leader, Dr. Alice (Farnham)")
combi$title2[797] <- "Mrs."
combi$title2 <- as.factor(combi$title2)
ggplot(combi[1:891,], aes(x = title2, fill = survived)) +
  geom_bar() +
```

```
  facet_wrap(~pclass) +
  ggtitle("Pclass") +
  xlab("Title") +
  ylab("Total Count") +
  labs(fill = "Survived")
combi$fsize <- combi$sibsp + combi$parch + 1
table(substr(combi$cabin, 1, 1))
n = nrow(train)
m = nrow(test)
train <- combi[1:n,]
test <- combi[(n+1):(n+m),]
library(party)
fit <- ctree(survived ~ pclass + sex + age + sibsp + parch, data =
train)
plot(fit)
yhat <- predict(fit,newdata=test,type="response")
table(yhat, test$survived)
mean(yhat == test$survived)
fit2 <- ctree(survived ~ pclass + title2 + fsize, data = train)
plot(fit2)
yhat2 <- predict(fit2,newdata=test,type="response")
table(yhat2, test$survived)
mean(yhat2 == test$survived)


#==========================
# spam
#==========================

spam <- read.csv("SPAM.csv",header=T)
spam$spam = as.factor(ifelse(spam$spam==T,"spam","email"))
train = spam[!spam$testid, -2]
test = spam[spam$testid, -2]
n = nrow(train)
m = nrow(test)
preds <- setdiff(colnames(train),"spam")
fml <- as.formula(paste("spam",
                  paste(preds,collapse=' + '),sep=' ~ '))
fml
score <- function(pred, truth, name="model") {
  ctable <- table(truth=truth,
                  pred=(pred>0.5))
  accuracy <- sum(diag(ctable))/sum(ctable)
  specificity <- ctable[1,1]/sum(ctable[1,])
  data.frame(model=name, accuracy=accuracy, specificity=specificity)
}
library(rpart)
fit.tree <- rpart(fml, train)
library(rpart.plot)
rpart.plot(fit.tree)
phat = predict(fit.tree, newdata=test)[,"spam"]
yhat = ifelse(phat>.5,"spam","email")
table(pred=yhat, truth=test$spam)
score(predict(fit.tree, newdata=test)[,"spam"], test$spam,
```

```
name="tree, test")
set.seed(123)
B <- 50
samples <- sapply(1:B,
                  FUN = function(iter)
                  {sample(1:n, size=n, replace=T)})
treelist <-lapply(1:B,
                  FUN=function(iter)
                  {samp <- samples[,iter];
                  rpart(fml, train[samp,])})
predict.bag <- function(treelist, newdata) {
  preds <- sapply(1:length(treelist),
                  FUN=function(iter) {
                  predict(treelist[[iter]], newdata=newdata)
[,"spam"]})
  predsums <- rowSums(preds)
  predsums/length(treelist)
}
score(predict.bag(treelist, newdata=test),
                 test$spam,
                 name="bagging, test")
library(randomForest)
set.seed(123)
p = ncol(test)
fit.rf <- randomForest(fml,
          data = train,
          ntree=B,
          mtry = sqrt(p),
          importance=T)
plot(fit.rf)
importance(fit.rf)[1:5,]
varImpPlot(fit.rf, type=1)
score(predict(fit.rf,
newdata=test[,preds],type='prob')[,"spam"],
test$spam, name="random forest, test")

#=========================
# rf
#=========================

library(MASS)
set.seed(123)
istrain = rbinom(n=nrow(Boston),size=1,prob=0.5)>0
train <- Boston[istrain,]
nrow(train)
test = Boston[!istrain,-14]
test.y = Boston[!istrain,14]
nrow(test)
library(randomForest)
set.seed(123)
fit1 = randomForest(medv ~ ., train, mtry=12, ntree=1)
yhat1 = predict(fit1, newdata=test)
fit2 = randomForest(medv ~ ., train, mtry=12, ntree=1)
yhat2 = predict(fit2, newdata=test)
```

```
plot(yhat1,yhat2)
abline(a=0,b=1)
cor(yhat1,yhat2)
fit1 = randomForest(medv ~ ., train, mtry=1, ntree=1)
yhat1 = predict(fit1, newdata=test)
fit2 = randomForest(medv ~ ., train, mtry=1, ntree=1)
yhat2 = predict(fit2, newdata=test)
plot(yhat1,yhat2)
abline(a=0,b=1)
cor(yhat1,yhat2)
set.seed(123)
fit <- randomForest(medv ~ ., train, importance=TRUE)
varImpPlot(fit)
MSE <- mean( ( predict(fit, newdata=test) - test.y )^2 )
test.perm = test
test.perm[,"rm"] = sample(test.perm[,"rm"])
MSE.perm <- mean( ( predict(fit, newdata=test.perm) - test.y )^2 )
MSE.perm - MSE
test.perm = test
test.perm[,"chas"] = sample(test.perm[,"chas"])
MSE.perm <- mean( ( predict(fit, newdata=test.perm) - test.y )^2 )
MSE.perm - MSE


#=========================
# L2boost
#=========================

load("poly250.Rdata")
set.seed(123)
y = f + rnorm(250,0,.01)
train = data.frame(x,y)
library(rpart)
B = 200+1
lambda = 0.1
d = 1
n = length(y)
r = y - mean(y)
fx = matrix(NA, nrow=n, ncol=B)
fx[,1] = rep(mean(y),n)
for (b in 2:B){
  fxb = rpart(r~x, control=rpart.control(maxdepth = d))
  fx[,b] = fx[,b-1] + lambda*predict(fxb)
  r = r - lambda*predict(fxb)
}
f.boost = fx[,B]
plot(x,f, col=2, type="l")
lines(x,f.boost, type="s", col="blue")
plot(x,r)
fxb = rpart(r~x, control=rpart.control(maxdepth = d))
lines(x,predict(fxb))


#=========================
```

```
# stacking
#=========================

library(MASS)
set.seed(123)
istrain = rbinom(n=nrow(Boston),size=1,prob=0.5)>0
train <- Boston[istrain,]
( n=nrow(train) )
test = Boston[!istrain,-14]
test.y = Boston[!istrain,14]
( m=nrow(test) )
fit1 = lm(medv ~ ., train)
attr(summary(fit1)$term,"term.labels")
library(rpart)
fit2 = rpart(medv ~ ., train)
z1 = vector()
z2 = z1
for (i in 1:n){
z1[i] = predict( lm(medv ~ (.), train[-i, ]),
                 newdata = train[i,]
                 )
z2[i] = predict(rpart(medv ~ ., train[-i,]),
                newdata = train[i,]
                )
}
fit = lm(medv ~ 0 + z1 + z2, train)
weights = coef(fit)
yhat1 = predict(fit1, newdata=test)
yhat2 = predict(fit2, newdata=test)
yhat = weights[1]*yhat1 + weights[2]*yhat2
cat("MSE stack: ", mean( (test.y - yhat)^2) )
cat("MSE lm: ", mean( (test.y - yhat1)^2) )
cat("MSE rpart: ", mean( (test.y - yhat2)^2) )


#=========================
# ridge
#=========================

require(ElemStatLearn)
train <- prostate[prostate$train,-10]
test <- prostate[!prostate$train,-10]
n <- nrow(train)
m <- nrow(test)
X<-as.matrix(
  scale(train[,-9])
  )
X.star <- as.matrix(
  scale(test[,-9])
  )
p = ncol(X)
y = scale(train[,"lpsa"])
y.star = scale(test[,"lpsa"])
lambda <- 10
```

```
hatbeta <- solve(t(X) %*% X + lambda*diag(p)) %*% t(X) %*% y
print(hatbeta)
lambdas = 0:10
nlambda = length(lambdas)
hatbetas <- matrix(NA,ncol=p, nrow=nlambda)
Vars <- matrix(NA,ncol=p, nrow=nlambda)
for (l in 1:nlambda){
  lambda = lambdas[l]
  hatbetas[l,] = solve(t(X) %*% X + lambda*diag(p)) %*% t(X) %*% y
  W = solve(diag(p) + lambda*solve(t(X) %*% X))
  Vars[l,] = diag( W%*% solve(t(X) %*% X) %*% t(W)  )
}
matplot(lambdas, hatbetas, type="l")
matplot(lambdas, Vars, type="l")
lambda = 5
LOOCV <- vector()
for (i in 1:n){
  X_i <- X[-i,]
  hatbeta_i = solve(t(X_i) %*% X_i + lambda*diag(p)) %*% t(X_i) %*%
y[-i]
  LOOCV[i] = ( X[i,] %*% hatbeta_i - y[i])^2
}
cat("LOOCV ridge: ", mean(LOOCV))
hatbeta <- solve(t(X) %*% X + lambda*diag(p)) %*% t(X) %*% y
yhat.ridge = X.star %*% hatbeta
cat("MSE ridge: ", mean((yhat.ridge - y.star)^2))
library(boot)
train.std = data.frame(y,X)
fit.lm = glm(y ~ 0 + ., train.std, family = gaussian)
cat("LOOCV lm: ", cv.glm(train.std, fit.lm)$delta[1])
yhat.lm = predict(fit.lm, newdata=data.frame(X.star))
cat("MSE lm: ", mean((yhat.lm - y.star)^2))


#=========================
# prostate
#=========================

require(ElemStatLearn)
train <- prostate[prostate$train,-10]
test <- prostate[!prostate$train,-10]
X = as.matrix(train[,-9])
y = train$lpsa
n<-nrow(X)
p<-ncol(X)
X.star = as.matrix(test[,-9])
y.star = test$lpsa
m = nrow(X.star)
library("corrplot")
corrplot(cor(train), method="number")
fit.full = lm(lpsa ~ ., train)
summary(fit.full)
mean( (predict(fit.full, newdata=test) - test$lpsa )^2 )
require(glmnet)
```

```
fit.ridge = glmnet(X,y,family="gaussian",alpha=0)
plot(fit.ridge, xvar="lambda")
K <- n
ridge.cv<-cv.glmnet(X,y,alpha=0, nfolds = K, grouped=FALSE)
plot(ridge.cv)
hatlambda <-ridge.cv$lambda.min
hatlambda
predict(fit.ridge, s=hatlambda, type ="coefficients")
plot(fit.ridge, xvar="lambda")
abline(v=log(hatlambda))
yhat.ridge = predict(fit.ridge, s=hatlambda, newx=X.star, exact=T)
mean( (yhat.ridge - test$lpsa)^2 )
library(penalized)
CVlik = profL2(y, penalized=X, minlambda2= 0.08, maxlambda2 = 800,
plot=T, trace=F, fold=1:n, standardize = T)
hatlambda = optL2(y,penalized=X, standardize = T, trace=F)$lambda
hatlambda
fit.ridge = penalized(y,penalized=X,lambda2=hatlambda, standardize =
T, trace=F)
yhat.ridge = predict(fit.ridge, penalized=X.star)[,1]
mean( (yhat.ridge - test$lpsa)^2 )
require(leaps)
fit.bests <- regsubsets(lpsa~.,train, nbest=1, nvmax=(p-1))
summary.bests<-summary(fit.bests)
summary.bests
names(summary.bests)
predict.regsubsets =function(object ,newdata ,id ,...){
 form=as.formula(object$call[[2]])
 mat=model.matrix(form, newdata)
 coefi =coef(object, id=id)
 xvars =names(coefi)
 mat[,xvars]%*%coefi
}
plot(summary.bests$cp, xlab="k", ylab="log Cp", type="b", log="y" )
kbest=which(summary.bests$cp==min(summary.bests$cp))
coef(fit.bests,kbest)
yhat.bestCp = predict.regsubsets(fit.bests, newdata=test, id=7)
mean( (yhat.bestCp - test$lpsa)^2 )
plot(fit.bests, scale="bic")
yhat.bestBIC = predict.regsubsets(fit.bests, newdata=test, id=2)
mean( (yhat.bestBIC - test$lpsa)^2 )
fit.null = lm(lpsa ~ 1, train)
fit.fwdAIC = step(fit.null, scope=list(upper=fit.full),
direction="forward", k=2, trace=0)
summary(fit.fwdAIC)
n*log(deviance(fit.fwdAIC)/n)+2*ncol(model.matrix(fit.fwdAIC))
yhat.fwdAIC = predict(fit.fwdAIC, newdata=test)
mean( (yhat.fwdAIC - test$lpsa)^2 )
set.seed(123)
K = 5
folds = sample(1:K, n, replace =TRUE)
KCV = matrix(NA, K, p)
for (k in 1:K){
fit_k = regsubsets(lpsa ~.,data=train[folds!=k,])
```

```
for (j in 1:p){
yhat_k=predict(fit_k, train[folds==k,], id=j)
KCV[k,j]=mean( (train$lpsa[folds==k]-yhat_k)^2 )
}
}
plot(1:p,apply(KCV,2,mean), type="b")
fit.lasso <- glmnet(X, y, alpha=1)
plot(fit.lasso, xvar="lambda")
set.seed(123)
K<-5
cv.lasso <-cv.glmnet(X, y, alpha=1, nfolds = K)
plot(cv.lasso)
hatlambda<-cv.lasso$lambda.1se
hatlambda
predict(fit.lasso ,s=hatlambda, type ="coefficients")
plot(fit.lasso, xvar="lambda")
abline(v=log(hatlambda))
yhat.lasso = predict(fit.lasso, s=hatlambda, newx=X.star, exact=T)
mean( (yhat.lasso - test$lpsa)^2 )


#==========================
# gam
#==========================

load("poly250.Rdata")
n = length(x)
set.seed(123)
sigma = 0.01
y = f + rnorm(n, 0, sigma)
train = data.frame(x=x,y=y)
plot(y~x, train)
lines(f ~ x, col=3)
require(splines)
knots = unique(train$x)
fit.o = smooth.spline(train$x, train$y, lambda = 1e-20, all.knots=T)
fit.o$lambda
fit.o$df
plot(y~x, train)
lines(f ~ x, col=3)
lines(train$x, predict(fit.o, x=train$x)$y, col=4)
fit.u = smooth.spline(train$x, train$y, lambda = 50, all.knots=T)
fit.u$lambda
fit.u$df
plot(y~x, train)
lines(f ~ x, col=3)
lines(train$x, predict(fit.u, x=train$x)$y, col=4)
fit = smooth.spline(train$x, train$y, cv=TRUE, all.knots=T)
fit$lambda
fit$df
plot(y~x, train)
lines(f ~ x, col=3)
lines(train$x, predict(fit, x=train$x)$y, col=4)
library(mgcv)
```

```
fit = gam(y ~ s(x), data=train)
summary(fit)
plot(fit, shade=TRUE)
library(ISLR)
fit = gam(wage ~ s(year, k=4) + s(age) + education, data=Wage)
summary(fit)
par(mfrow =c(1,3))
plot(fit, shade=TRUE, all.terms = TRUE)
fit = gam(wage ~ s(year, age) + education, data=Wage)
vis.gam(fit, view = c("year","age"),theta= -135)


#=========================
# CVwrong
#=========================

set.seed(123)
n = 50
p = 5000
y = c(rep(0,n/2),rep(1,n/2))
X = matrix(rnorm(p*n), ncol=p)
cors = apply(X,2, function(x) cor(y,x))
colbest100 = sort(-abs(cors), index.return=T)$ix[1:100]
Xbest = X[,colbest100]
require(class)
K<-5
set.seed(123)
folds <- sample( rep(1:K,length=n) )
Err.CV = vector()
for (k in 1:K){
out = which(folds==k)
pred <- knn(train = Xbest[ -out, ],
            test = Xbest[out, ],
            cl = y[-out], k = 1)
Err.CV[k] = mean( y[out] != pred)
}
cat("5-fold CV:", mean(Err.CV))
set.seed(123)
n = 50
p = 5000
y = c(rep(0,n/2),rep(1,n/2))
X = matrix(rnorm(p*n), ncol=p)
require(class)
K<-5
set.seed(123)
folds <- sample( rep(1:K,length=n) )
Err.CV = vector()
for (k in 1:K){
out = which(folds==k)
cors = apply(X[-out, ],2, function(x) cor(y[-out],x))
colbest100 = sort(-abs(cors), index.return=T)$ix[1:100]
Xbest = X[,colbest100]
pred <- knn(train = Xbest[-out, ],
            test = Xbest[out, ],
```

```
            cl = y[-out], k = 1)

    # % of misclassified samples
Err.CV[k] = mean( y[out] != pred)
}
cat("5-fold CV:", mean(Err.CV))


#=========================
# ISL
#=========================

set.seed(1)
y=rnorm(100)
x=rnorm(100)
y=x-2* x^2+ rnorm(100)
train = data.frame(y,x)
library(boot)
LOOCV <- vector()
for (d in 1:4){
LOOCV[d] = cv.glm(train, glm(y ~ poly(x,d))  )$delta[1]
}
names(LOOCV) <- c("i","ii","iii","iv")
LOOCV
set.seed(123)
p = 20
n = 1000
X = matrix(rnorm(p*n), ncol=p)
beta = c(rep(1,p/4),rep(0,3*p/4))
y = X%*%beta + rnorm(n)
combi = data.frame(y=y, X=X)
train = combi[1:100,]
test = combi[101:1000,]
library(leaps)
fit <- regsubsets(y ~ .,train, nvmax=p)
plot(1:p, summary(fit)$rss/n, log="y")
predict.regsubsets =function(object ,newdata ,id ,...){
form=as.formula(object$call[[2]])
mat=model.matrix(form, newdata)
coefi =coef(object, id=id)
xvars =names(coefi)
mat[,xvars]%*%coefi
}
MSE.te = vector()
for (k in 1:p){
yhat = predict.regsubsets(fit, newdata=test, id=k)
MSE.te[k] = mean( (yhat - test$y)^2 )
}
plot(1:p,MSE.te, log="y")
= which.min(MSE.te)
k
hatbetas = matrix(0,ncol=21,nrow=20)
 where = summary(fit)$which
 beta1 = c(0,beta)
```

```
 diff = vector()
 for (k in 1:20){
 hatbetas[k,where[k,]] <- coef(fit,k)
 diff[k] <- sqrt( sum( (hatbetas[k,] - beta1)^2 )  )
 }
plot(1:p,diff)
set.seed(123)
n = 100
x1 = rnorm(n)
x2 = rnorm(n)
y = 5+ 1*x1 + 3*x2 + rnorm(n)
hatbeta1 = -5
hatbeta1
r = y - hatbeta1*x1
hatbeta2 = lm(r ~ x2)$coef[2]
hatbeta2
r = y - hatbeta2*x2
hatbeta1 = lm(r ~ x1)$coef[2]
hatbeta1
B=10
hatbetas = matrix(0,ncol=3,nrow=B)
hatbetas[1,2] <- -5
for (b in 2:B){
r = y - hatbetas[b-1,2]*x1
hatbetas[b,3] = lm(r ~ x2)$coef[2]
r = y - hatbetas[b,3]*x2
hatbetas[b,2] = lm(r ~ x1)$coef[2]
hatbetas[b,1] = lm(r ~ x1)$coef[1]
}
matplot(hatbetas, type="l")
fit = lm(y ~ x1 +x2)
matplot(hatbetas, type="l")
abline(h=coef(fit), lty=3)
```