```
#==================================
# An Introduction to Statistical Learning
# with Applications in R
# Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
# R code for Labs
#==================================

# Chapter 2 Lab: Introduction to R

# Basic Commands

x <- c(1,3,2,5)
x
x = c(1,6,2)
x
y = c(1,4,3)
length(x)
length(y)
x+y
ls()
rm(x,y)
ls()
rm(list=ls())
?matrix
x=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
x
x=matrix(c(1,2,3,4),2,2)
matrix(c(1,2,3,4),2,2,byrow=TRUE)
sqrt(x)
x^2
x=rnorm(50)
y=x+rnorm(50,mean=50,sd=.1)
cor(x,y)
set.seed(1303)
rnorm(50)
set.seed(3)
y=rnorm(100)
mean(y)
var(y)
sqrt(var(y))
sd(y)

# Graphics

x=rnorm(100)
y=rnorm(100)
plot(x,y)
plot(x,y,xlab="this is the x-axis",ylab="this is the y-
axis",main="Plot of X vs Y")
pdf("Figure.pdf")
plot(x,y,col="green")
dev.off()
x=seq(1,10)
x
```

```
x=1:10
x
x=seq(-pi,pi,length=50)
y=x
f=outer(x,y,function(x,y)cos(y)/(1+x^2))
contour(x,y,f)
contour(x,y,f,nlevels=45,add=T)
fa=(f-t(f))/2
contour(x,y,fa,nlevels=15)
image(x,y,fa)
persp(x,y,fa)
persp(x,y,fa,theta=30)
persp(x,y,fa,theta=30,phi=20)
persp(x,y,fa,theta=30,phi=70)
persp(x,y,fa,theta=30,phi=40)

# Indexing Data

A=matrix(1:16,4,4)
A
A[2,3]
A[c(1,3),c(2,4)]
A[1:3,2:4]
A[1:2,]
A[,1:2]
A[1,]
A[-c(1,3),]
A[-c(1,3),-c(1,3,4)]
dim(A)

# Loading Data

Auto=read.table("Auto.data")
fix(Auto)
Auto=read.table("Auto.data",header=T,na.strings="?")
fix(Auto)
Auto=read.csv("Auto.csv",header=T,na.strings="?")
fix(Auto)
dim(Auto)
Auto[1:4,]
Auto=na.omit(Auto)
dim(Auto)
names(Auto)

# Additional Graphical and Numerical Summaries

plot(cylinders, mpg)
plot(Auto$cylinders, Auto$mpg)
attach(Auto)
plot(cylinders, mpg)
cylinders=as.factor(cylinders)
plot(cylinders, mpg)
plot(cylinders, mpg, col="red")
plot(cylinders, mpg, col="red", varwidth=T)
```

```
plot(cylinders, mpg, col="red", varwidth=T,horizontal=T)
plot(cylinders, mpg, col="red", varwidth=T, xlab="cylinders",
ylab="MPG")
hist(mpg)
hist(mpg,col=2)
hist(mpg,col=2,breaks=15)
pairs(Auto)
pairs(~ mpg + displacement + horsepower + weight + acceleration,
Auto)
plot(horsepower,mpg)
identify(horsepower,mpg,name)
summary(Auto)
summary(mpg)



# Chapter 3 Lab: Linear Regression

library(MASS)
library(ISLR)

# Simple Linear Regression

fix(Boston)
names(Boston)
lm.fit=lm(medv~lstat)
lm.fit=lm(medv~lstat,data=Boston)
attach(Boston)
lm.fit=lm(medv~lstat)
lm.fit
summary(lm.fit)
names(lm.fit)
coef(lm.fit)
confint(lm.fit)
predict(lm.fit,data.frame(lstat=(c(5,10,15))),
interval="confidence")
predict(lm.fit,data.frame(lstat=(c(5,10,15))),
interval="prediction")
plot(lstat,medv)
abline(lm.fit)
abline(lm.fit,lwd=3)
abline(lm.fit,lwd=3,col="red")
plot(lstat,medv,col="red")
plot(lstat,medv,pch=20)
plot(lstat,medv,pch="+")
plot(1:20,1:20,pch=1:20)
par(mfrow=c(2,2))
plot(lm.fit)
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
plot(hatvalues(lm.fit))
which.max(hatvalues(lm.fit))

# Multiple Linear Regression
```

```
lm.fit=lm(medv~lstat+age,data=Boston)
summary(lm.fit)
lm.fit=lm(medv~.,data=Boston)
summary(lm.fit)
library(car)
vif(lm.fit)
lm.fit1=lm(medv~.-age,data=Boston)
summary(lm.fit1)
lm.fit1=update(lm.fit, ~.-age)

# Interaction Terms

summary(lm(medv~lstat*age,data=Boston))

# Non-linear Transformations of the Predictors

lm.fit2=lm(medv~lstat+I(lstat^2))
summary(lm.fit2)
lm.fit=lm(medv~lstat)
anova(lm.fit,lm.fit2)
par(mfrow=c(2,2))
plot(lm.fit2)
lm.fit5=lm(medv~poly(lstat,5))
summary(lm.fit5)
summary(lm(medv~log(rm),data=Boston))

# Qualitative Predictors

fix(Carseats)
names(Carseats)
lm.fit=lm(Sales~.+Income:Advertising+Price:Age,data=Carseats)
summary(lm.fit)
attach(Carseats)
contrasts(ShelveLoc)

# Writing Functions

LoadLibraries
LoadLibraries()
LoadLibraries=function(){
 library(ISLR)
 library(MASS)
 print("The libraries have been loaded.")
 }
LoadLibraries
LoadLibraries()


# Chapter 4 Lab: Logistic Regression, LDA, QDA, and KNN

# The Stock Market Data

library(ISLR)
```

```
names(Smarket)
dim(Smarket)
summary(Smarket)
pairs(Smarket)
cor(Smarket)
cor(Smarket[,-9])
attach(Smarket)
plot(Volume)

# Logistic Regression

glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,f
amily=binomial)
summary(glm.fit)
coef(glm.fit)
summary(glm.fit)$coef
summary(glm.fit)$coef[,4]
glm.probs=predict(glm.fit,type="response")
glm.probs[1:10]
contrasts(Direction)
glm.pred=rep("Down",1250)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction)
(507+145)/1250
mean(glm.pred==Direction)
train=(Year<2005)
Smarket.2005=Smarket[!train,]
dim(Smarket.2005)
Direction.2005=Direction[!train]
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,f
amily=binomial,subset=train)
glm.probs=predict(glm.fit,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)
mean(glm.pred==Direction.2005)
mean(glm.pred!=Direction.2005)
glm.fit=glm(Direction~Lag1+Lag2,data=Smarket,family=binomial,subset=
train)
glm.probs=predict(glm.fit,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)
mean(glm.pred==Direction.2005)
106/(106+76)
predict(glm.fit,newdata=data.frame(Lag1=c(1.2,1.5),Lag2=c(1.1,-0.8))
,type="response")

# Linear Discriminant Analysis

library(MASS)
lda.fit=lda(Direction~Lag1+Lag2,data=Smarket,subset=train)
lda.fit
plot(lda.fit)
```

```
lda.pred=predict(lda.fit, Smarket.2005)
names(lda.pred)
lda.class=lda.pred$class
table(lda.class,Direction.2005)
mean(lda.class==Direction.2005)
sum(lda.pred$posterior[,1]>=.5)
sum(lda.pred$posterior[,1]<.5)
lda.pred$posterior[1:20,1]
lda.class[1:20]
sum(lda.pred$posterior[,1]>.9)

# Quadratic Discriminant Analysis

qda.fit=qda(Direction~Lag1+Lag2,data=Smarket,subset=train)
qda.fit
qda.class=predict(qda.fit,Smarket.2005)$class
table(qda.class,Direction.2005)
mean(qda.class==Direction.2005)

# K-Nearest Neighbors

library(class)
train.X=cbind(Lag1,Lag2)[train,]
test.X=cbind(Lag1,Lag2)[!train,]
train.Direction=Direction[train]
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=1)
table(knn.pred,Direction.2005)
(83+43)/252
knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,Direction.2005)
mean(knn.pred==Direction.2005)

# An Application to Caravan Insurance Data

dim(Caravan)
attach(Caravan)
summary(Purchase)
348/5822
standardized.X=scale(Caravan[,-86])
var(Caravan[,1])
var(Caravan[,2])
var(standardized.X[,1])
var(standardized.X[,2])
test=1:1000
train.X=standardized.X[-test,]
test.X=standardized.X[test,]
train.Y=Purchase[-test]
test.Y=Purchase[test]
set.seed(1)
knn.pred=knn(train.X,test.X,train.Y,k=1)
mean(test.Y!=knn.pred)
mean(test.Y!="No")
table(knn.pred,test.Y)
```

```
9/(68+9)
knn.pred=knn(train.X,test.X,train.Y,k=3)
table(knn.pred,test.Y)
5/26
knn.pred=knn(train.X,test.X,train.Y,k=5)
table(knn.pred,test.Y)
4/15
glm.fit=glm(Purchase~.,data=Caravan,family=binomial,subset=-test)
glm.probs=predict(glm.fit,Caravan[test,],type="response")
glm.pred=rep("No",1000)
glm.pred[glm.probs>.5]="Yes"
table(glm.pred,test.Y)
glm.pred=rep("No",1000)
glm.pred[glm.probs>.25]="Yes"
table(glm.pred,test.Y)
11/(22+11)



# Chaper 5 Lab: Cross-Validation and the Bootstrap

# The Validation Set Approach

library(ISLR)
set.seed(1)
train=sample(392,196)
lm.fit=lm(mpg~horsepower,data=Auto,subset=train)
attach(Auto)
mean((mpg-predict(lm.fit,Auto))[-train]^2)
lm.fit2=lm(mpg~poly(horsepower,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
lm.fit3=lm(mpg~poly(horsepower,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)
set.seed(2)
train=sample(392,196)
lm.fit=lm(mpg~horsepower,subset=train)
mean((mpg-predict(lm.fit,Auto))[-train]^2)
lm.fit2=lm(mpg~poly(horsepower,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
lm.fit3=lm(mpg~poly(horsepower,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)

# Leave-One-Out Cross-Validation

glm.fit=glm(mpg~horsepower,data=Auto)
coef(glm.fit)
lm.fit=lm(mpg~horsepower,data=Auto)
coef(lm.fit)
library(boot)
glm.fit=glm(mpg~horsepower,data=Auto)
cv.err=cv.glm(Auto,glm.fit)
cv.err$delta
cv.error=rep(0,5)
for (i in 1:5){
```

```
 glm.fit=glm(mpg~poly(horsepower,i),data=Auto)
 cv.error[i]=cv.glm(Auto,glm.fit)$delta[1]
 }
cv.error

# k-Fold Cross-Validation

set.seed(17)
cv.error.10=rep(0,10)
for (i in 1:10){
 glm.fit=glm(mpg~poly(horsepower,i),data=Auto)
 cv.error.10[i]=cv.glm(Auto,glm.fit,K=10)$delta[1]
 }
cv.error.10

# The Bootstrap

alpha.fn=function(data,index){
 X=data$X[index]
 Y=data$Y[index]
 return((var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y)))
 }
alpha.fn(Portfolio,1:100)
set.seed(1)
alpha.fn(Portfolio,sample(100,100,replace=T))
boot(Portfolio,alpha.fn,R=1000)

# Estimating the Accuracy of a Linear Regression Model

boot.fn=function(data,index)
 return(coef(lm(mpg~horsepower,data=data,subset=index)))
boot.fn(Auto,1:392)
set.seed(1)
boot.fn(Auto,sample(392,392,replace=T))
boot.fn(Auto,sample(392,392,replace=T))
boot(Auto,boot.fn,1000)
summary(lm(mpg~horsepower,data=Auto))$coef
boot.fn=function(data,index)

coefficients(lm(mpg~horsepower+I(horsepower^2),data=data,subset=inde
x))
set.seed(1)
boot(Auto,boot.fn,1000)
summary(lm(mpg~horsepower+I(horsepower^2),data=Auto))$coef



# Chapter 6 Lab 1: Subset Selection Methods

# Best Subset Selection

library(ISLR)
fix(Hitters)
names(Hitters)
```

```
dim(Hitters)
sum(is.na(Hitters$Salary))
Hitters=na.omit(Hitters)
dim(Hitters)
sum(is.na(Hitters))
library(leaps)
regfit.full=regsubsets(Salary~.,Hitters)
summary(regfit.full)
regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=19)
reg.summary=summary(regfit.full)
names(reg.summary)
reg.summary$rsq
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted
RSq",type="l")
which.max(reg.summary$adjr2)
points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
which.min(reg.summary$cp)
points(10,reg.summary$cp[10],col="red",cex=2,pch=20)
which.min(reg.summary$bic)
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(6,reg.summary$bic[6],col="red",cex=2,pch=20)
plot(regfit.full,scale="r2")
plot(regfit.full,scale="adjr2")
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
coef(regfit.full,6)

# Forward and Backward Stepwise Selection

regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward
")
summary(regfit.fwd)
regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="backwar
d")
summary(regfit.bwd)
coef(regfit.full,7)
coef(regfit.fwd,7)
coef(regfit.bwd,7)

# Choosing Among Models

set.seed(1)
train=sample(c(TRUE,FALSE), nrow(Hitters),rep=TRUE)
test=(!train)
regfit.best=regsubsets(Salary~.,data=Hitters[train,],nvmax=19)
test.mat=model.matrix(Salary~.,data=Hitters[test,])
val.errors=rep(NA,19)
for(i in 1:19){
   coefi=coef(regfit.best,id=i)
   pred=test.mat[,names(coefi)]%*%coefi
   val.errors[i]=mean((Hitters$Salary[test]-pred)^2)
```

```
}
val.errors
which.min(val.errors)
coef(regfit.best,10)
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
  }
regfit.best=regsubsets(Salary~.,data=Hitters,nvmax=19)
coef(regfit.best,10)
k=10
set.seed(1)
folds=sample(1:k,nrow(Hitters),replace=TRUE)
cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))
for(j in 1:k){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
  for(i in 1:19){
    pred=predict(best.fit,Hitters[folds==j,],id=i)
    cv.errors[j,i]=mean( (Hitters$Salary[folds==j]-pred)^2)
    }
  }
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
reg.best=regsubsets(Salary~.,data=Hitters, nvmax=19)
coef(reg.best,11)


# Chapter 6 Lab 2: Ridge Regression and the Lasso

x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary

# Ridge Regression

library(glmnet)
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
ridge.mod$lambda[50]
coef(ridge.mod)[,50]
sqrt(sum(coef(ridge.mod)[-1,50]^2))
ridge.mod$lambda[60]
coef(ridge.mod)[,60]
sqrt(sum(coef(ridge.mod)[-1,60]^2))
predict(ridge.mod,s=50,type="coefficients")[1:20,]
set.seed(1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
```

```
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid,
thresh=1e-12)
ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
mean((ridge.pred-y.test)^2)
mean((mean(y[train])-y.test)^2)
ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])
mean((ridge.pred-y.test)^2)
ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T)
mean((ridge.pred-y.test)^2)
lm(y~x, subset=train)
predict(ridge.mod,s=0,exact=T,type="coefficients")[1:20,]
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlam)[1:20,]

# The Lasso

lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test)^2)
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]
lasso.coef
lasso.coef[lasso.coef!=0]


# Chapter 6 Lab 3: PCR and PLS Regression

# Principal Components Regression

library(pls)
set.seed(2)
pcr.fit=pcr(Salary~., data=Hitters,scale=TRUE,validation="CV")
summary(pcr.fit)
validationplot(pcr.fit,val.type="MSEP")
set.seed(1)
pcr.fit=pcr(Salary~., data=Hitters,subset=train,scale=TRUE,
validation="CV")
validationplot(pcr.fit,val.type="MSEP")
pcr.pred=predict(pcr.fit,x[test,],ncomp=7)
mean((pcr.pred-y.test)^2)
pcr.fit=pcr(y~x,scale=TRUE,ncomp=7)
summary(pcr.fit)
```

```
# Partial Least Squares

set.seed(1)
pls.fit=plsr(Salary~., data=Hitters,subset=train,scale=TRUE,
validation="CV")
summary(pls.fit)
validationplot(pls.fit,val.type="MSEP")
pls.pred=predict(pls.fit,x[test,],ncomp=2)
mean((pls.pred-y.test)^2)
pls.fit=plsr(Salary~., data=Hitters,scale=TRUE,ncomp=2)
summary(pls.fit)



# Chapter 7 Lab: Non-linear Modeling

library(ISLR)
attach(Wage)

# Polynomial Regression and Step Functions

fit=lm(wage~poly(age,4),data=Wage)
coef(summary(fit))
fit2=lm(wage~poly(age,4,raw=T),data=Wage)
coef(summary(fit2))
fit2a=lm(wage~age+I(age^2)+I(age^3)+I(age^4),data=Wage)
coef(fit2a)
fit2b=lm(wage~cbind(age,age^2,age^3,age^4),data=Wage)
agelims=range(age)
age.grid=seq(from=agelims[1],to=agelims[2])
preds=predict(fit,newdata=list(age=age.grid),se=TRUE)
se.bands=cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit)
par(mfrow=c(1,2),mar=c(4.5,4.5,1,1),oma=c(0,0,4,0))
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
title("Degree-4 Polynomial",outer=T)
lines(age.grid,preds$fit,lwd=2,col="blue")
matlines(age.grid,se.bands,lwd=1,col="blue",lty=3)
preds2=predict(fit2,newdata=list(age=age.grid),se=TRUE)
max(abs(preds$fit-preds2$fit))
fit.1=lm(wage~age,data=Wage)
fit.2=lm(wage~poly(age,2),data=Wage)
fit.3=lm(wage~poly(age,3),data=Wage)
fit.4=lm(wage~poly(age,4),data=Wage)
fit.5=lm(wage~poly(age,5),data=Wage)
anova(fit.1,fit.2,fit.3,fit.4,fit.5)
coef(summary(fit.5))
(-11.983)^2
fit.1=lm(wage~education+age,data=Wage)
fit.2=lm(wage~education+poly(age,2),data=Wage)
fit.3=lm(wage~education+poly(age,3),data=Wage)
anova(fit.1,fit.2,fit.3)
fit=glm(I(wage>250)~poly(age,4),data=Wage,family=binomial)
preds=predict(fit,newdata=list(age=age.grid),se=T)
```

```
pfit=exp(preds$fit)/(1+exp(preds$fit))
se.bands.logit = cbind(preds$fit+2*preds$se.fit,
preds$fit-2*preds$se.fit)
se.bands = exp(se.bands.logit)/(1+exp(se.bands.logit))
preds=predict(fit,newdata=list(age=age.grid),type="response",se=T)
plot(age,I(wage>250),xlim=agelims,type="n",ylim=c(0,.2))
points(jitter(age), I((wage>250)/5),cex=.5,pch="|",col="darkgrey")
lines(age.grid,pfit,lwd=2, col="blue")
matlines(age.grid,se.bands,lwd=1,col="blue",lty=3)
table(cut(age,4))
fit=lm(wage~cut(age,4),data=Wage)
coef(summary(fit))

# Splines

library(splines)
fit=lm(wage~bs(age,knots=c(25,40,60)),data=Wage)
pred=predict(fit,newdata=list(age=age.grid),se=T)
plot(age,wage,col="gray")
lines(age.grid,pred$fit,lwd=2)
lines(age.grid,pred$fit+2*pred$se,lty="dashed")
lines(age.grid,pred$fit-2*pred$se,lty="dashed")
dim(bs(age,knots=c(25,40,60)))
dim(bs(age,df=6))
attr(bs(age,df=6),"knots")
fit2=lm(wage~ns(age,df=4),data=Wage)
pred2=predict(fit2,newdata=list(age=age.grid),se=T)
lines(age.grid, pred2$fit,col="red",lwd=2)
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
title("Smoothing Spline")
fit=smooth.spline(age,wage,df=16)
fit2=smooth.spline(age,wage,cv=TRUE)
fit2$df
lines(fit,col="red",lwd=2)
lines(fit2,col="blue",lwd=2)
legend("topright",legend=c("16 DF","6.8
DF"),col=c("red","blue"),lty=1,lwd=2,cex=.8)
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey")
title("Local Regression")
fit=loess(wage~age,span=.2,data=Wage)
fit2=loess(wage~age,span=.5,data=Wage)
lines(age.grid,predict(fit,data.frame(age=age.grid)),col="red",lwd=2
)
lines(age.grid,predict(fit2,data.frame(age=age.grid)),col="blue",lwd
=2)
legend("topright",legend=c("Span=0.2","Span=0.5"),col=c("red","blue"
),lty=1,lwd=2,cex=.8)

# GAMs

gam1=lm(wage~ns(year,4)+ns(age,5)+education,data=Wage)
library(gam)
gam.m3=gam(wage~s(year,4)+s(age,5)+education,data=Wage)
par(mfrow=c(1,3))
```

```
plot(gam.m3, se=TRUE,col="blue")
plot.gam(gam1, se=TRUE, col="red")
gam.m1=gam(wage~s(age,5)+education,data=Wage)
gam.m2=gam(wage~year+s(age,5)+education,data=Wage)
anova(gam.m1,gam.m2,gam.m3,test="F")
summary(gam.m3)
preds=predict(gam.m2,newdata=Wage)
gam.lo=gam(wage~s(year,df=4)+lo(age,span=0.7)+education,data=Wage)
plot.gam(gam.lo, se=TRUE, col="green")
gam.lo.i=gam(wage~lo(year,age,span=0.5)+education,data=Wage)
library(akima)
plot(gam.lo.i)
gam.lr=gam(I(wage>250)~year+s(age,df=5)+education,family=binomial,da
ta=Wage)
par(mfrow=c(1,3))
plot(gam.lr,se=T,col="green")
table(education,I(wage>250))
gam.lr.s=gam(I(wage>250)~year+s(age,df=5)+education,family=binomial,
data=Wage,subset=(education!="1. < HS Grad"))
plot(gam.lr.s,se=T,col="green")



# Chapter 8 Lab: Decision Trees

# Fitting Classification Trees

library(tree)
library(ISLR)
attach(Carseats)
High=ifelse(Sales<=8,"No","Yes")
Carseats=data.frame(Carseats,High)
tree.carseats=tree(High~.-Sales,Carseats)
summary(tree.carseats)
plot(tree.carseats)
text(tree.carseats,pretty=0)
tree.carseats
set.seed(2)
train=sample(1:nrow(Carseats), 200)
Carseats.test=Carseats[-train,]
High.test=High[-train]
tree.carseats=tree(High~.-Sales,Carseats,subset=train)
tree.pred=predict(tree.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
(86+57)/200
set.seed(3)
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
names(cv.carseats)
cv.carseats
par(mfrow=c(1,2))
plot(cv.carseats$size,cv.carseats$dev,type="b")
plot(cv.carseats$k,cv.carseats$dev,type="b")
prune.carseats=prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
```

```
text(prune.carseats,pretty=0)
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
(94+60)/200
prune.carseats=prune.misclass(tree.carseats,best=15)
plot(prune.carseats)
text(prune.carseats,pretty=0)
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
(86+62)/200

# Fitting Regression Trees

library(MASS)
set.seed(1)
train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
plot(tree.boston)
text(tree.boston,pretty=0)
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
prune.boston=prune.tree(tree.boston,best=5)
plot(prune.boston)
text(prune.boston,pretty=0)
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test)
abline(0,1)
mean((yhat-boston.test)^2)

# Bagging and Random Forests

library(randomForest)
set.seed(1)
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,impo
rtance=TRUE)
bag.boston
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
mean((yhat.bag-boston.test)^2)
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntre
e=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,import
ance=TRUE)
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
importance(rf.boston)
varImpPlot(rf.boston)
```

```
# Boosting

library(gbm)
set.seed(1)
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",
n.trees=5000,interaction.depth=4)
summary(boost.boston)
par(mfrow=c(1,2))
plot(boost.boston,i="rm")
plot(boost.boston,i="lstat")
yhat.boost=predict(boost.boston,newdata=Boston[-
train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",
n.trees=5000,interaction.depth=4,shrinkage=0.2,verbose=F)
yhat.boost=predict(boost.boston,newdata=Boston[-
train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)



# Chapter 9 Lab: Support Vector Machines

# Support Vector Classifier

set.seed(1)
x=matrix(rnorm(20*2), ncol=2)
y=c(rep(-1,10), rep(1,10))
x[y==1,]=x[y==1,] + 1
plot(x, col=(3-y))
dat=data.frame(x=x, y=as.factor(y))
library(e1071)
svmfit=svm(y~., data=dat, kernel="linear", cost=10,scale=FALSE)
plot(svmfit, dat)
svmfit$index
summary(svmfit)
svmfit=svm(y~., data=dat, kernel="linear", cost=0.1,scale=FALSE)
plot(svmfit, dat)
svmfit$index
set.seed(1)
tune.out=tune(svm,y~.,data=dat,kernel="linear",ranges=list(cost=c(0.
001, 0.01, 0.1, 1,5,10,100)))
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
xtest=matrix(rnorm(20*2), ncol=2)
ytest=sample(c(-1,1), 20, rep=TRUE)
xtest[ytest==1,]=xtest[ytest==1,] + 1
testdat=data.frame(x=xtest, y=as.factor(ytest))
ypred=predict(bestmod,testdat)
table(predict=ypred, truth=testdat$y)
svmfit=svm(y~., data=dat, kernel="linear", cost=.01,scale=FALSE)
ypred=predict(svmfit,testdat)
table(predict=ypred, truth=testdat$y)
```

```
x[y==1,]=x[y==1,]+0.5
plot(x, col=(y+5)/2, pch=19)
dat=data.frame(x=x,y=as.factor(y))
svmfit=svm(y~., data=dat, kernel="linear", cost=1e5)
summary(svmfit)
plot(svmfit, dat)
svmfit=svm(y~., data=dat, kernel="linear", cost=1)
summary(svmfit)
plot(svmfit,dat)

# Support Vector Machine

set.seed(1)
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
train=sample(200,100)
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=1, cost=1)
plot(svmfit, dat[train,])
summary(svmfit)
svmfit=svm(y~., data=dat[train,], kernel="radial",gamma=1,cost=1e5)
plot(svmfit,dat[train,])
set.seed(1)
tune.out=tune(svm, y~., data=dat[train,], kernel="radial",
ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))
summary(tune.out)
table(true=dat[-train,"y"],
pred=predict(tune.out$best.model,newx=dat[-train,]))

# ROC Curves

library(ROCR)
rocplot=function(pred, truth, ...){
   predob = prediction(pred, truth)
   perf = performance(predob, "tpr", "fpr")
   plot(perf,...)}
svmfit.opt=svm(y~., data=dat[train,], kernel="radial",gamma=2,
cost=1,decision.values=T)
fitted=attributes(predict(svmfit.opt,dat[train,],decision.values=TRU
E))$decision.values
par(mfrow=c(1,2))
rocplot(fitted,dat[train,"y"],main="Training Data")
svmfit.flex=svm(y~., data=dat[train,], kernel="radial",gamma=50,
cost=1, decision.values=T)
fitted=attributes(predict(svmfit.flex,dat[train,],decision.values=T)
)$decision.values
rocplot(fitted,dat[train,"y"],add=T,col="red")
fitted=attributes(predict(svmfit.opt,dat[-
train,],decision.values=T))$decision.values
rocplot(fitted,dat[-train,"y"],main="Test Data")
fitted=attributes(predict(svmfit.flex,dat[-
```

```
train,],decision.values=T))$decision.values
rocplot(fitted,dat[-train,"y"],add=T,col="red")

# SVM with Multiple Classes

set.seed(1)
x=rbind(x, matrix(rnorm(50*2), ncol=2))
y=c(y, rep(0,50))
x[y==0,2]=x[y==0,2]+2
dat=data.frame(x=x, y=as.factor(y))
par(mfrow=c(1,1))
plot(x,col=(y+1))
svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)
plot(svmfit, dat)

# Application to Gene Expression Data

library(ISLR)
names(Khan)
dim(Khan$xtrain)
dim(Khan$xtest)
length(Khan$ytrain)
length(Khan$ytest)
table(Khan$ytrain)
table(Khan$ytest)
dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))
out=svm(y~., data=dat, kernel="linear",cost=10)
summary(out)
table(out$fitted, dat$y)
dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))
pred.te=predict(out, newdata=dat.te)
table(pred.te, dat.te$y)



# Chapter 10 Lab 1: Principal Components Analysis

states=row.names(USArrests)
states
names(USArrests)
apply(USArrests, 2, mean)
apply(USArrests, 2, var)
pr.out=prcomp(USArrests, scale=TRUE)
names(pr.out)
pr.out$center
pr.out$scale
pr.out$rotation
dim(pr.out$x)
biplot(pr.out, scale=0)
pr.out$rotation=-pr.out$rotation
pr.out$x=-pr.out$x
biplot(pr.out, scale=0)
pr.out$sdev
pr.var=pr.out$sdev^2
```

```
pr.var
pve=pr.var/sum(pr.var)
pve
plot(pve, xlab="Principal Component", ylab="Proportion of Variance
Explained", ylim=c(0,1),type='b')
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative
Proportion of Variance Explained", ylim=c(0,1),type='b')
a=c(1,2,8,-3)
cumsum(a)


# Chapter 10 Lab 2: Clustering

# K-Means Clustering

set.seed(2)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
km.out=kmeans(x,2,nstart=20)
km.out$cluster
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results
with K=2", xlab="", ylab="", pch=20, cex=2)
set.seed(4)
km.out=kmeans(x,3,nstart=20)
km.out
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results
with K=3", xlab="", ylab="", pch=20, cex=2)
set.seed(3)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss

# Hierarchical Clustering

hc.complete=hclust(dist(x), method="complete")
hc.average=hclust(dist(x), method="average")
hc.single=hclust(dist(x), method="single")
par(mfrow=c(1,3))
plot(hc.complete,main="Complete Linkage", xlab="", sub="", cex=.9)
plot(hc.average, main="Average Linkage", xlab="", sub="", cex=.9)
plot(hc.single, main="Single Linkage", xlab="", sub="", cex=.9)
cutree(hc.complete, 2)
cutree(hc.average, 2)
cutree(hc.single, 2)
cutree(hc.single, 4)
xsc=scale(x)
plot(hclust(dist(xsc), method="complete"), main="Hierarchical
Clustering with Scaled Features")
x=matrix(rnorm(30*3), ncol=3)
dd=as.dist(1-cor(t(x)))
plot(hclust(dd, method="complete"), main="Complete Linkage with
Correlation-Based Distance", xlab="", sub="")
```

```
# Chapter 10 Lab 3: NCI60 Data Example

# The NCI60 data

library(ISLR)
nci.labs=NCI60$labs
nci.data=NCI60$data
dim(nci.data)
nci.labs[1:4]
table(nci.labs)

# PCA on the NCI60 Data

pr.out=prcomp(nci.data, scale=TRUE)
Cols=function(vec){
    cols=rainbow(length(unique(vec)))
    return(cols[as.numeric(as.factor(vec))])
  }
par(mfrow=c(1,2))
plot(pr.out$x[,1:2], col=Cols(nci.labs), pch=19,xlab="Z1",ylab="Z2")
plot(pr.out$x[,c(1,3)], col=Cols(nci.labs),
pch=19,xlab="Z1",ylab="Z3")
summary(pr.out)
plot(pr.out)
pve=100*pr.out$sdev^2/sum(pr.out$sdev^2)
par(mfrow=c(1,2))
plot(pve,  type="o", ylab="PVE", xlab="Principal Component",
col="blue")
plot(cumsum(pve), type="o", ylab="Cumulative PVE", xlab="Principal
Component", col="brown3")

# Clustering the Observations of the NCI60 Data

sd.data=scale(nci.data)
par(mfrow=c(1,3))
data.dist=dist(sd.data)
plot(hclust(data.dist), labels=nci.labs, main="Complete Linkage",
xlab="", sub="",ylab="")
plot(hclust(data.dist, method="average"), labels=nci.labs,
main="Average Linkage", xlab="", sub="",ylab="")
plot(hclust(data.dist, method="single"), labels=nci.labs,
main="Single Linkage", xlab="", sub="",ylab="")
hc.out=hclust(dist(sd.data))
hc.clusters=cutree(hc.out,4)
table(hc.clusters,nci.labs)
par(mfrow=c(1,1))
plot(hc.out, labels=nci.labs)
abline(h=139, col="red")
hc.out
set.seed(2)
km.out=kmeans(sd.data, 4, nstart=20)
km.clusters=km.out$cluster
```

```
table(km.clusters,hc.clusters)
hc.out=hclust(dist(pr.out$x[,1:5]))
plot(hc.out, labels=nci.labs, main="Hier. Clust. on First Five Score
Vectors")
table(cutree(hc.out,4), nci.labs)
```