Study Notes inspired by the: Statistics for Risk Modeling (SRM) SOA exam syllabus[1]

Terence Lim[2]
terence@alum.mit.edu

Last Updated: December 2019

[1]see Society of Actuaries (SOA) educational pathway https://pathways.soa.org

Contents

Textbooks

[REG] *Regression Modeling with Actuarial and Financial Applications*, Edward W. Frees, 2010, New York: Cambridge. ISBN: 978-0521135962.

[ISL] *An Introduction to Statistical Learning, with Applications in R*, James, Witten, Hastie, Tibshirani, 2013, New York: Springer.
A PDF version of the textbook can be downloaded at www.statlearning.com
R code for labs retrieved from http://faculty.marshall.usc.edu/gareth-james/ISL/

Other References

R code from labs in the textbook were run in RStudio and included as Markdown reports, a suggested reference (from the SOA PA syllabus) is: *R for Everyone*, 2nd ed. Lander, 2017, Boston: Addison-Wesley, ISBN 978-0-13-454692-6.

Many learning topic questions were inspired by the FDP Institute Study Guide, retrieved from
https://fdpinstitute.org/Study-Guide

1. Topic: Basics of Statistical Learning

The Candidate will understand key concepts of statistical learning

1. Explain the types of modeling problems and methods, including supervised versus unsupervised learning and regression versus classification.
2. Explain the common methods of assessing model accuracy.
3. Employ basic methods of exploratory data analysis, including data checking and validation

1.1 Statistical learning ([ISL] Ch. 2.1)

Explain why we estimate a function with data, including the role of input and output variables and their synonyms, as well as error terms (reducible and irreducible), expected values and variance.

There are two main reasons:

1. Prediction. When inputs X are readily available and the error term averages to zero, but the output Y cannot be easily obtained, we can predict Y using our estimate of the function $\hat{Y} = \hat{f}(X)$. The accuracy of $\hat{Y}$ as a prediction for Y depends on the reducible error and the irreducible error. We can potentially improve the accuracy of $\hat{f}$ by using the most appropriate statistical learning technique to reduce the error. estimate f . However, if Y is also a function of an error term ϵ (which tains unmeasured variables useful in predicting Y), then this is known as the irreducible error, because no matter how well we estimate f, we cannot reduce the error. We can show that the expected (or average) squared different between the predicted and actual value is:

$$\begin{aligned} E[(Y - \hat{Y})^2] &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= [f(X) - \hat{f}(X)]^2 + Var(\epsilon) \\ &= \text{Reducible Error} + \text{Irreducible Error} \end{aligned}$$

   where $Var(\epsilon)$ represents the variance associated with the error term.

2. Inference. We instead want to understand the relationship between X and Y:
   - Which predictors are associated with the response?
   - What is the relationship between the response and each predictor?
   - Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

Compare and contrast parametric and non-parametric learning methods.
Parametric methods involve a two-step model-based approach:

1. First, we make an assumption about the functional form, or shape, of f (e.g. that f is linear in X)
2. After a model has been selected, we need a procedure that uses the training data to fit or train the model (e.g. finding the linear coefficients by the ordinary least squares approach).

Non-parametric methods do not make explicit assumptions about the functional form of f . Instead they seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly. By avoiding the assumption of a particular functional form for f, they have the potential to accurately fit a wider range of possible shapes for f.

Describe the trade-offs between prediction accuracy, flexibility and model interpretability, including the role of overfitting.
In general, as the flexibility of a method increases, its interpretability decreases. Surprisingly, even if we are only interested in prediction, more accurate predictions are often obtained by a less flexible method, due to the potential for overfitting in highly flexible methods.

Determine when a supervised learning model is preferable to unsupervised or semi-supervised learning models.
When this is an associated response measurement for each observation of the predictor measurement(s), supervised learning can fit a model that relates the response to the predictors, with the aim of accurately predicting the response for future observations (prediction) or better understanding the relationship between the response and the predictors (inference). In the somewhat more challenging situation in which for every observation, a vector of measurements but no associated response are observed, then unsupervised learning is used to understand the relationships between the variables or observations. When response measurements are only available for some of the observations, semi-supervised learnings methods are used which incorporate observations for which response measurements are available as well observations for which they are not.

Explain how the appropriateness of regression problems relative to classification problems may be related to whether responses are quantitative or qualitative.
Variables can be characterized as either quantitative take on numerical values, or qualitative (also known as categorical) which take on values in one of K different classes, or categories. Problems with a quantitative response tend to be referred to as regression problems, while those involving a qualitative response as classification problems.

1.2 Assessing Model Accuracy ([ISL] Ch. 2.2)

Define predictors
Predictors are variables, other than the response variable, that are measured of each observation.

Recognize and explain the equation for mean squared error.
In the regression setting, the most commonly-used measure of model accuracy is the mean squared error: $MSE = \frac{1}{n}\sum_i^n (y_i - \hat{f}(x_i)^2$, where $\hat{f}(x_i)$ is the prediction for the ith observation. MSE measures the average squared difference of the predicted and true responses. It quantifies the extent to which the predicted response value for a given observation is close to the true response value for that observation.The MSE

will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.

Explain the goal of measuring the quality of fit by minimizing training and test mean square errors (MSEs) and the implications of different levels of flexibility (degrees of freedom) for both training and test MSEs.

We are really interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen test data. The problem is that many statistical methods specifically estimate coefficients so as to minimize the training set MSE, but the test MSE is often much larger. The degrees of freedom is a quantity that summarizes the flexibility of a function. As model flexibility increases, training MSE will decrease, but the test MSE may not.

Explain expected test MSE and the purpose of cross validation.

MSE measures the average squared difference of the predicted and true responses. It quantifies the extent to which the predicted response value for a given observation is close to the true response value for that observation.In practice, one can usually compute the training MSE with relative ease, but estimating test MSE is considerably more difficult because usually no test data are available. Cross-validation, is a method for estimating test MSE using the training data.

Define bias

Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.

Explain the bias-variance trade-off with an MSE decomposition into three fundamental quantities.

Expected test MSE can always be decomposed into the sum of three fundamental quantities: variance of $\hat{f}(x)$, the squared bias of $\hat{f}(x)$ and the variance of the error term ϵ.

$$E[y - \hat{f}(x)]^2 = \text{Var}(\hat{f}(x) + [\text{Bias}(\hat{f}(x))]^2 + \text{Var}(\epsilon)$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. The challenge, or trade-off, lies in finding a method for which both the variance and the squared bias are low.

Define indicator variable

An indicator variabletakes on a value of 1 or 0, and measures if an observation was classified correctly by our classification method; otherwise it was misclassified.

Explain the salient features of a simple Bayes classifier (for two classes) including conditional probability, the Bayes decision boundary and Bayes error rate.

The simple Bayes classifier assigns each observation to the most likely class, given its predictor values. In a two-class problem, this corresponds to predicting class one if the conditional probability $Pr(Y = 1|X = x)) > 0.5$, and class two otherwise. The Bayes decision boundary, which is plotted in the multi-dimensional space consisting of the predictors X, represents the set of points for which the probabilities are 50%. The Bayes classifier produces the lowest possible test error rate, called the Bayes error rate.The Bayes error rate is analogous to the irreducible error,

Explain how the K-nearest neighbors classifier is related to the Bayes classifier and how the choice of K impacts results.

The KNN classifier first identifies the K points in the training data that are closest to x. It then estimates the conditional probability for class j as the fraction of points whose response values equal j. Finally,

KNN applies Bayes rule and classifies the test observation to the class with the largest probability. As K grows, the method becomes less flexible and produces a decision boundary that is close to linear. This corresponds to a low-variance but high-bias classifier.

1.3 Lab: Introduction to R ([ISL] Ch 2.3)


```r
# Basic Commands
x <- c(1,3,2,5)
x
```

```
## [1] 1 3 2 5
```

```r
x = c(1,6,2)
x
```

```
## [1] 1 6 2
```

```r
y = c(1,4,3)
length(x)
```

```
## [1] 3
```

```r
length(y)
```

```
## [1] 3
```

```r
x+y
```

```
## [1]  2 10  5
```

```r
ls()
```

```
## [1] "x" "y"
```

```r
rm(x,y)
ls()
```

```
## character(0)
```

```r
rm(list=ls())
?matrix
x=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
x
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```r
x=matrix(c(1,2,3,4),2,2)
matrix(c(1,2,3,4),2,2,byrow=TRUE)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

```r
sqrt(x)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```r
x^2
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

```r
x=rnorm(50)
y=x+rnorm(50,mean=50,sd=.1)
cor(x,y)
```

```
## [1] 0.9930185
```

```r
set.seed(1303)
rnorm(50)
```

```
##  [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665
##  [6]  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890 -1.1102250073
## [11] -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091
## [16] -0.5563104914 -0.3647543571  0.8623550343 -0.6307715354  0.3136021252
## [21] -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099
## [31]  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917
## [36]  2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065
## [41]  0.2640073322  0.6321868074 -1.3306509858  0.0268888182  1.0406363208
## [46]  1.3120237985 -0.0300020767 -0.2500257125  0.0234144857  1.6598706557
```

```r
set.seed(3)
y=rnorm(100)
mean(y)
```

```
## [1] 0.01103557
```

```r
var(y)
```

```
## [1] 0.7328675
```

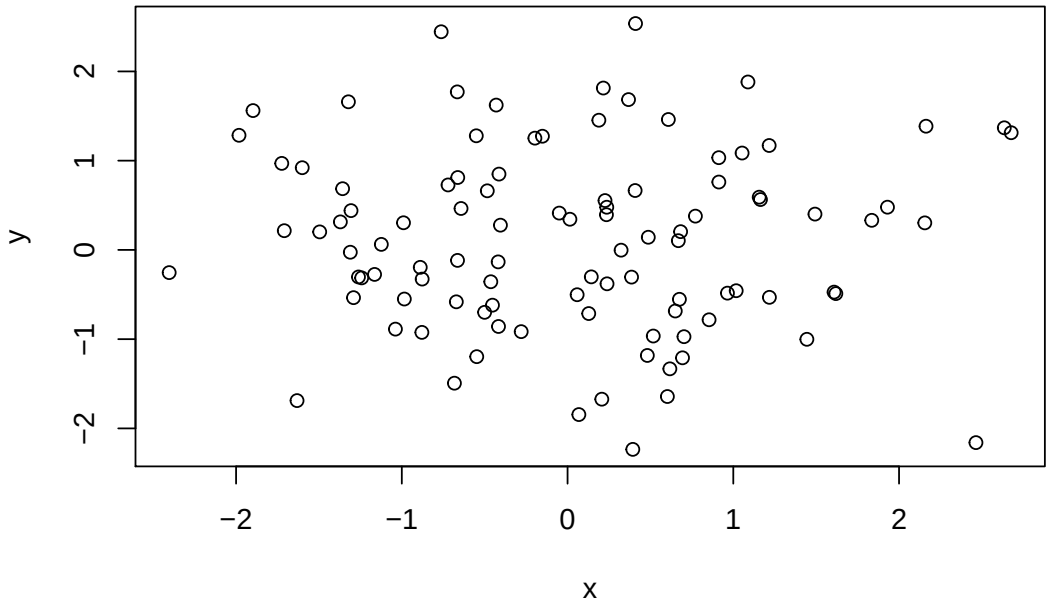
```r
sqrt(var(y))
```

```
## [1] 0.8560768
```

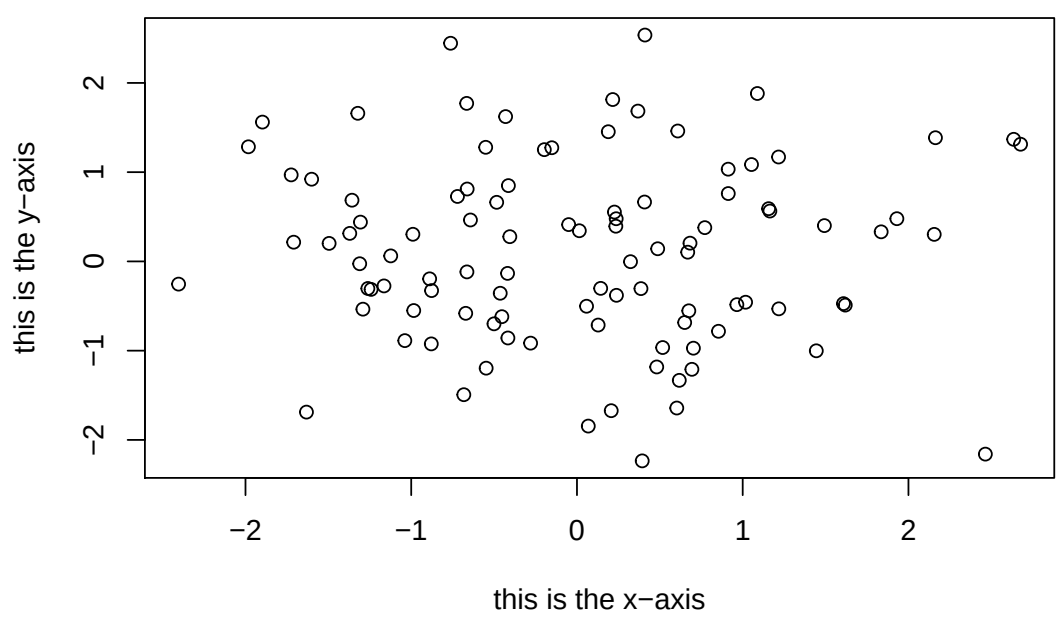
```r
sd(y)
```

```
## [1] 0.8560768
```

```r
# 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()
```

```
## pdf
##   2
```

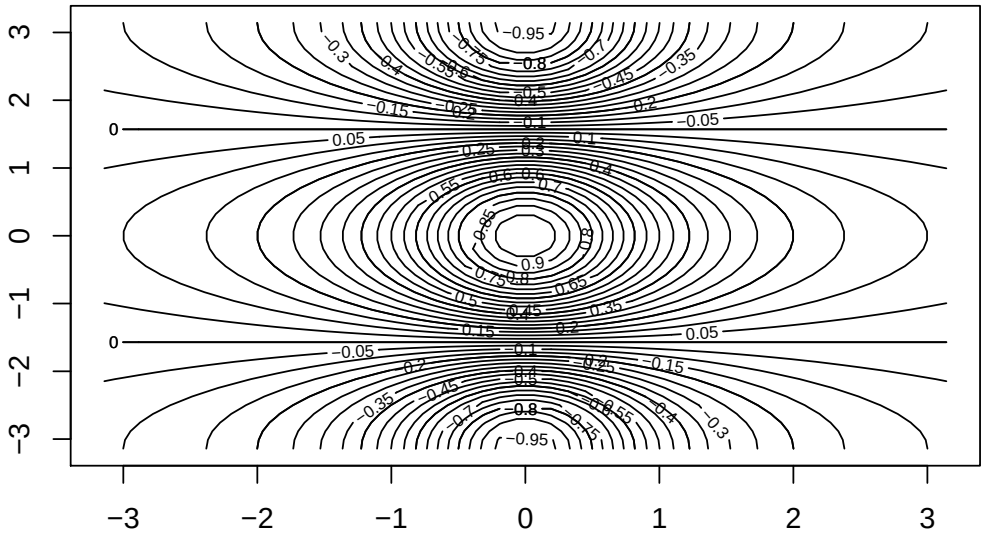
```
x=seq(1,10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

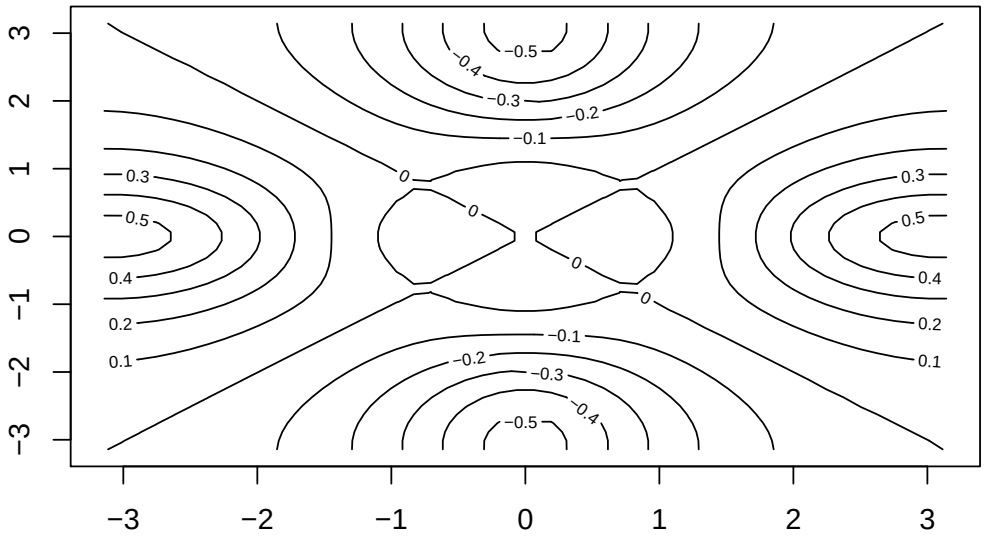
```
x=1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

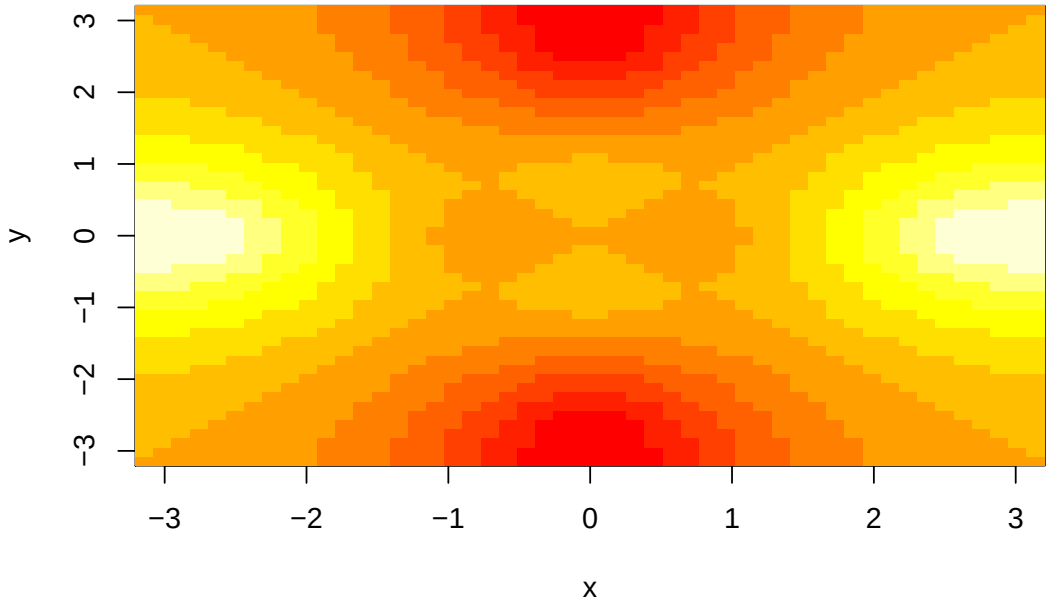
```
x=seq(-pi,pi,length=50)
y=x
f=outer(x,y,function(x,y)cos(y)/(1+x^2))
contour(x,y,f)
contour(x,y,f,nlevels=45,add=T)
```



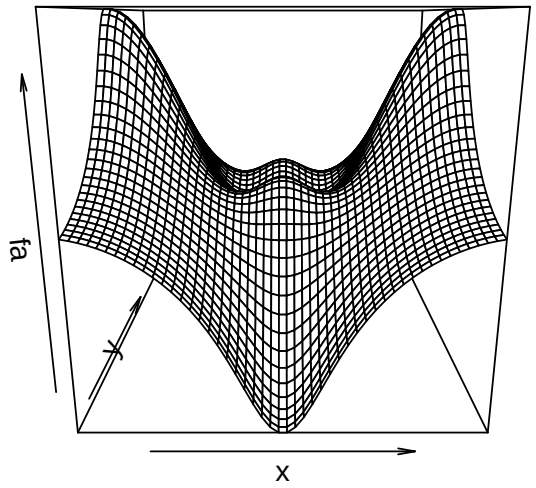

```
fa=(f-t(f))/2
contour(x,y,fa,nlevels=15)
```



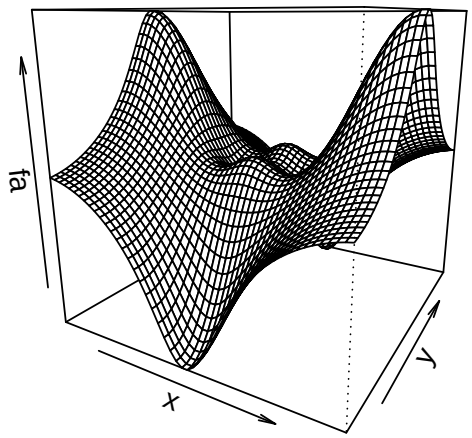

```
image(x,y,fa)
```



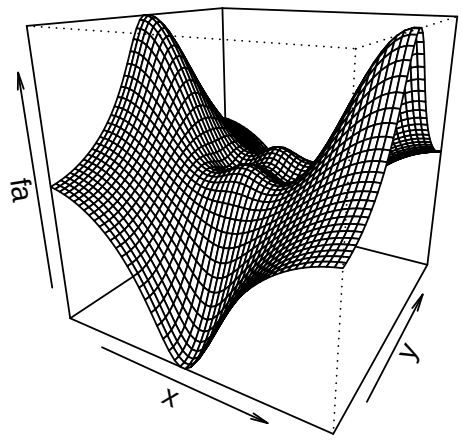

```
persp(x,y,fa)
```



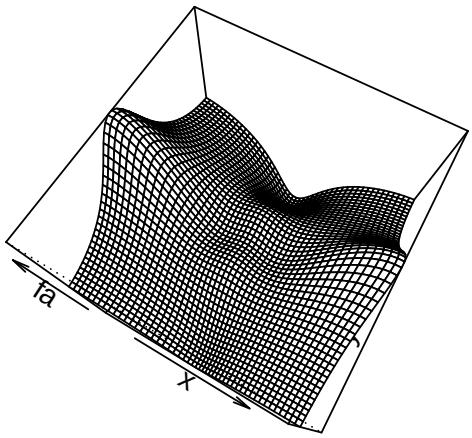

```
persp(x,y,fa,theta=30)
```



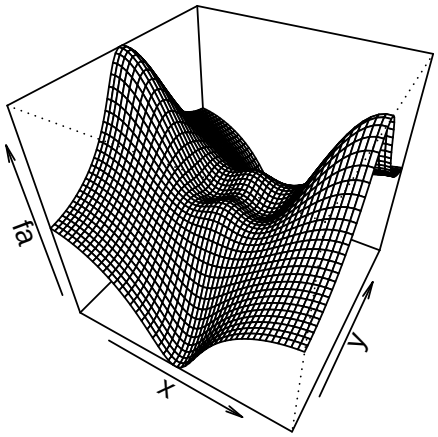

```
persp(x,y,fa,theta=30,phi=20)
```




```
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
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
A[2,3]
```

```
## [1] 10
```

```
A[c(1,3),c(2,4)]
```

```
##      [,1] [,2]
## [1,]    5   13
## [2,]    7   15
```

```
A[1:3,2:4]
```

```
##      [,1] [,2] [,3]
```

```
## [1,]    5    9   13
## [2,]    6   10   14
## [3,]    7   11   15
```

```
A[1:2,]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
```

```
A[,1:2]
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

```
A[1,]
```

```
## [1]  1  5  9 13
```

```
A[-c(1,3),]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    6   10   14
## [2,]    4    8   12   16
```

```
A[-c(1,3),-c(1,3,4)]
```

```
## [1] 6 8
```

```
dim(A)
```

```
## [1] 4 4
```

```
# Loading Data
library(ISLR)  # collection of data-sets used in the book
# 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)
```

```
## [1] 392   9
```

```
Auto[1:4,]
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
```

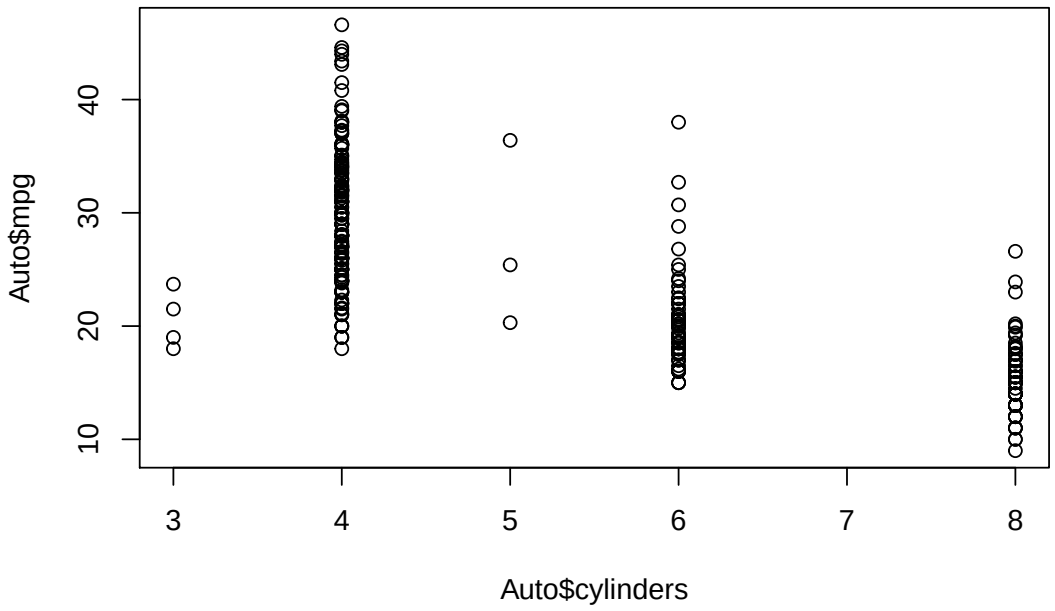
```r
Auto=na.omit(Auto)
dim(Auto)
```

```
## [1] 392   9
```

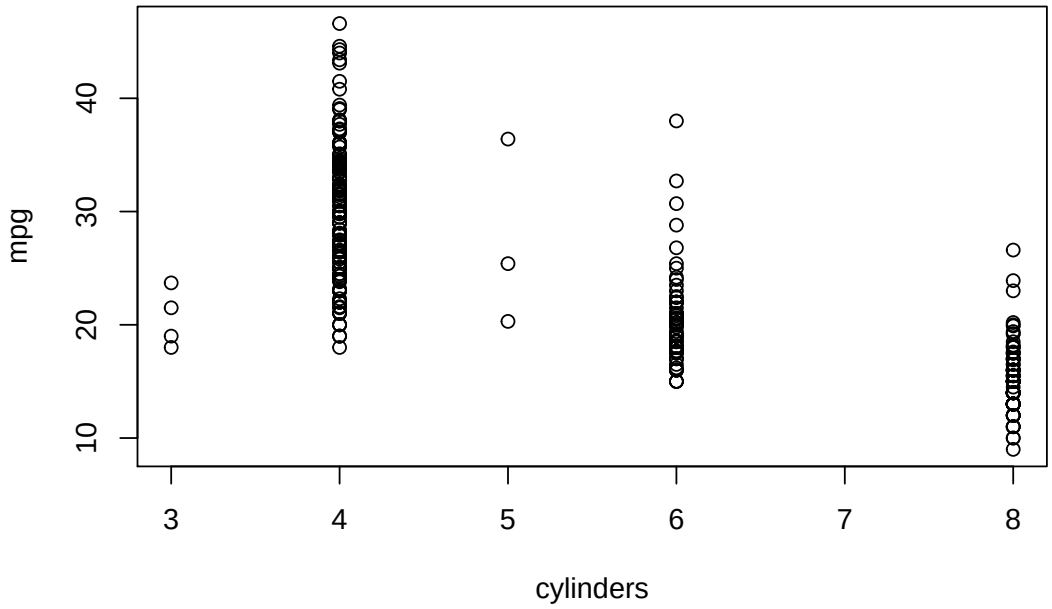
```r
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"
## [5] "weight"       "acceleration" "year"         "origin"
## [9] "name"
```

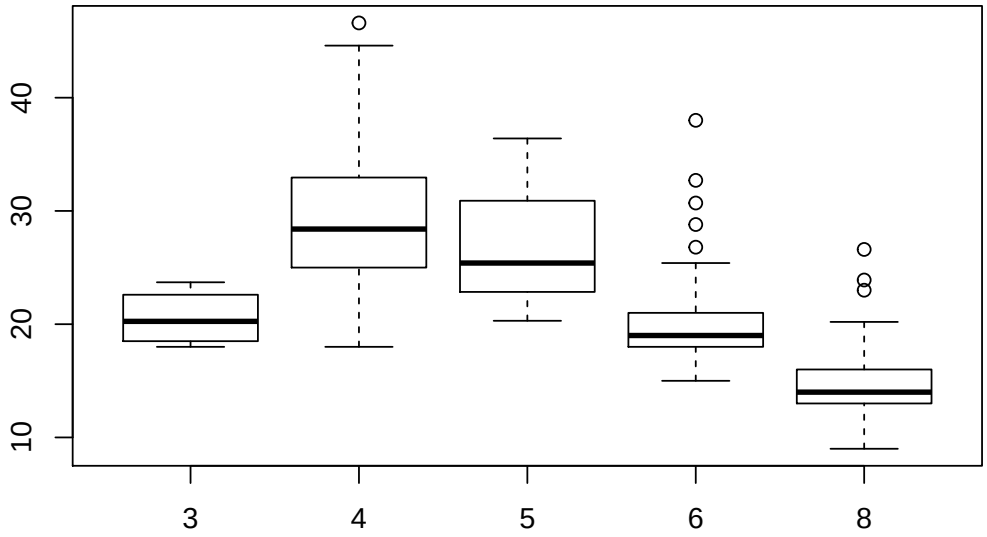
```r
# Additional Graphical and Numerical Summaries

# plot(cylinders, mpg) # Error in plot ( cylinders , mpg ) : object ' cylinders ' not found
plot(Auto$cylinders, Auto$mpg)
```



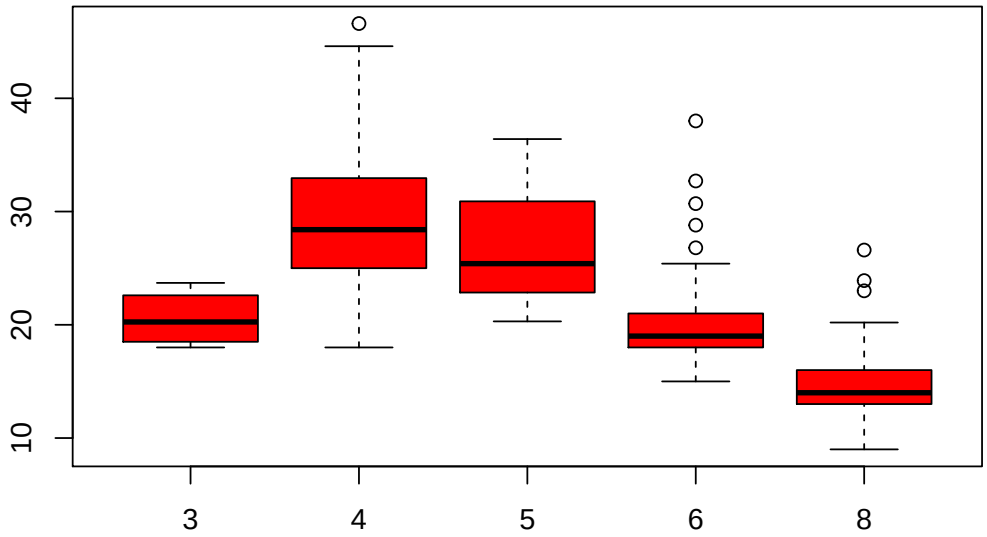

```r
attach(Auto)
plot(cylinders, mpg)
```



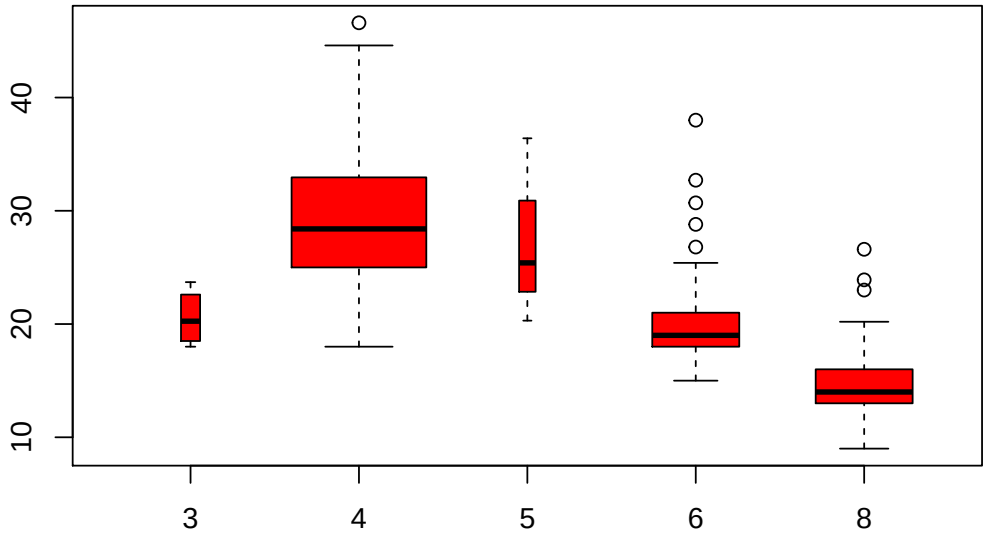

```
cylinders=as.factor(cylinders)
plot(cylinders, mpg)
```



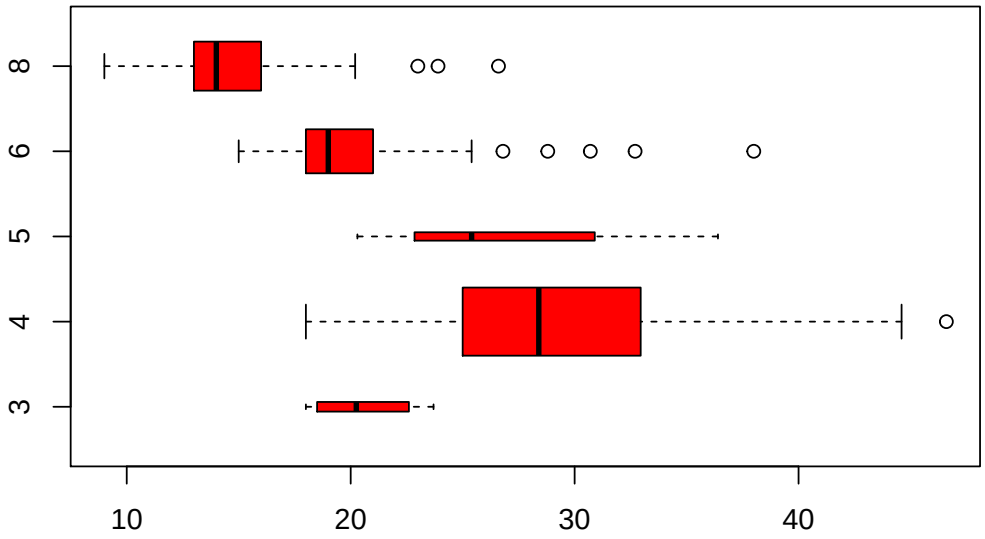

```
plot(cylinders, mpg, col="red")
```



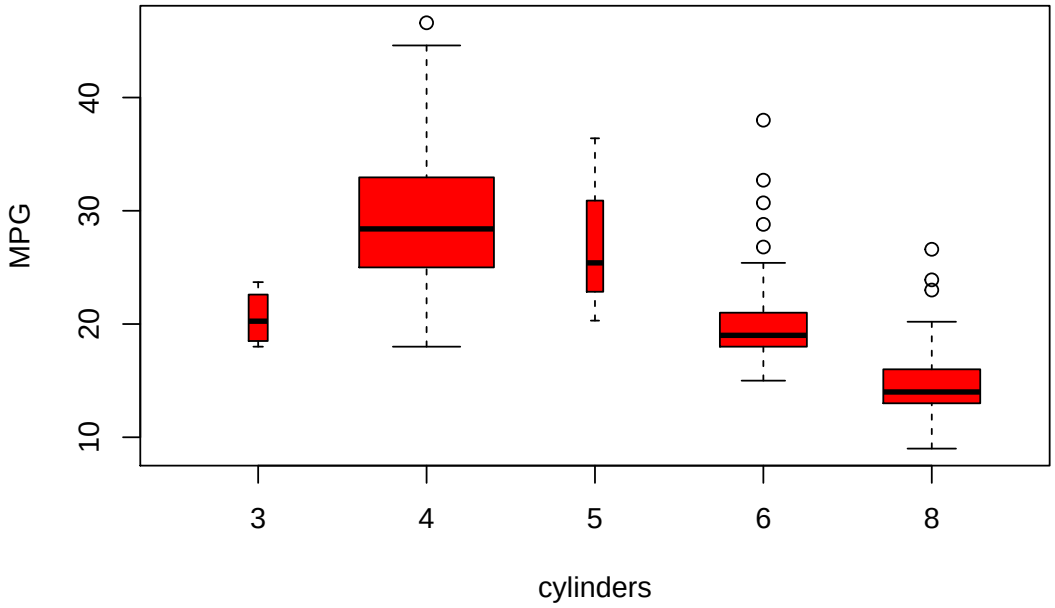

```
plot(cylinders, mpg, col="red", varwidth=T)
```



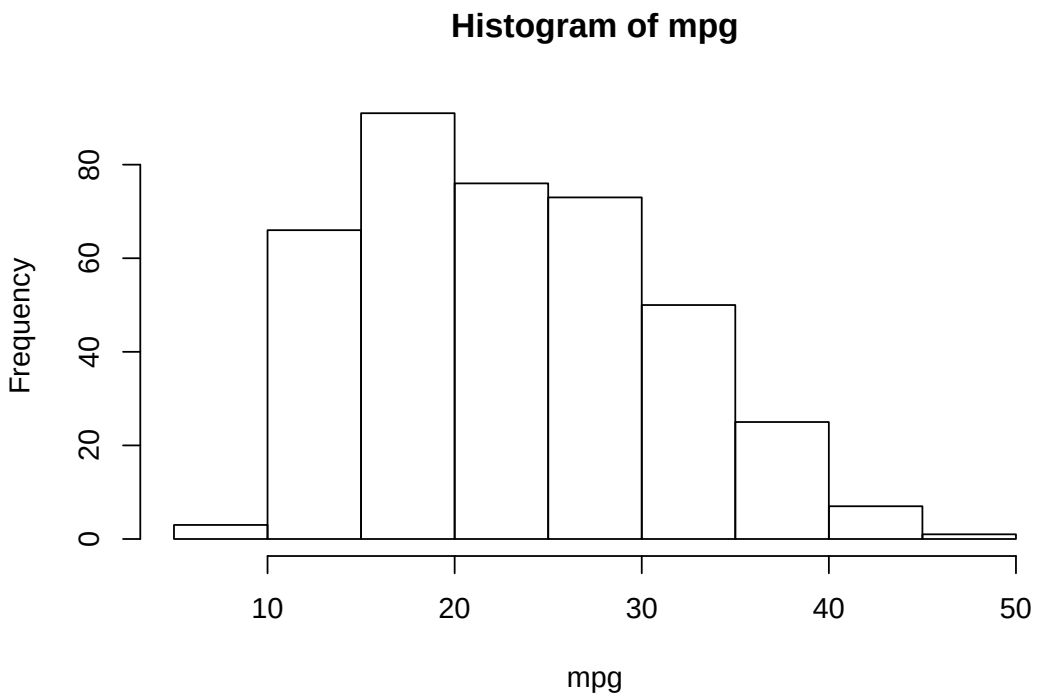

```
plot(cylinders, mpg, col="red", varwidth=T,horizontal=T)
```



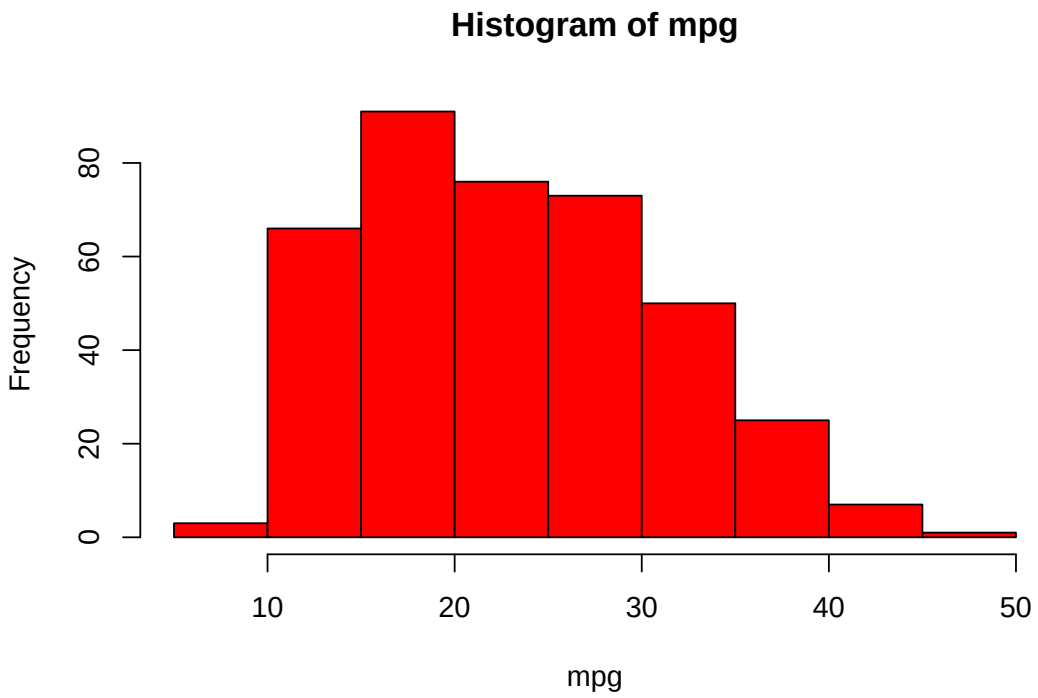

```
plot(cylinders, mpg, col="red", varwidth=T, xlab="cylinders", ylab="MPG")
```



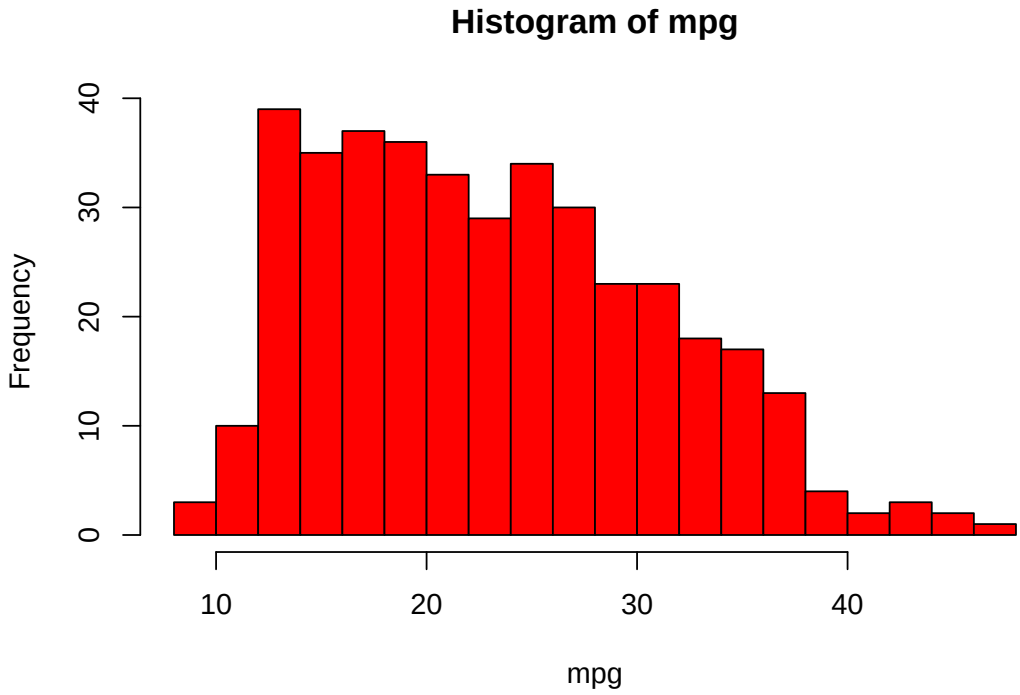

```
hist(mpg)
```



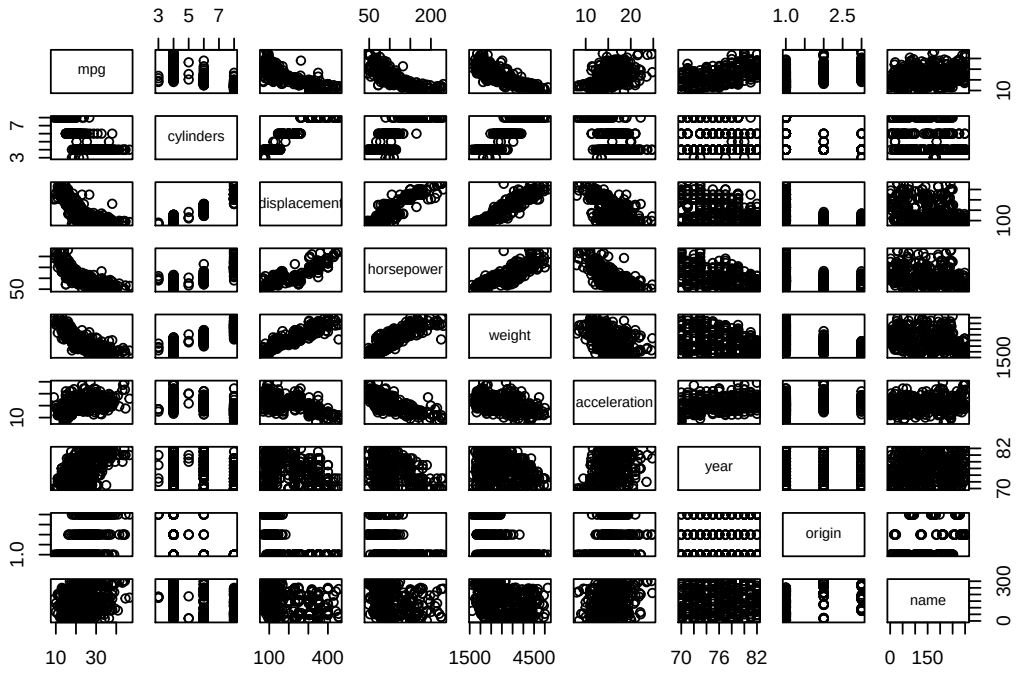

```
hist(mpg,col=2)
```



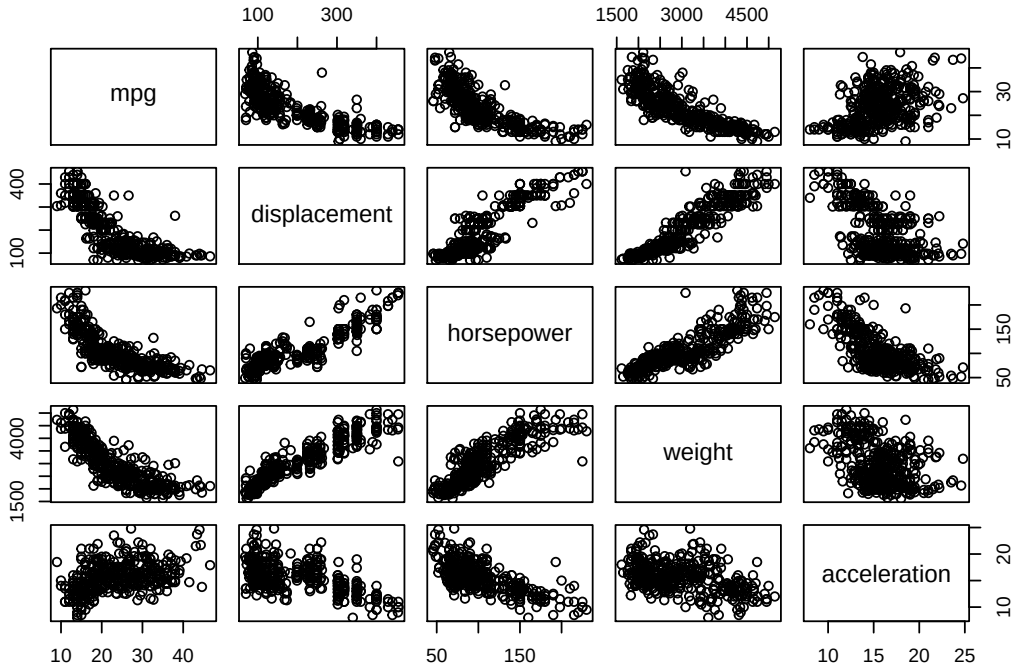

```
hist(mpg,col=2,breaks=15)
```



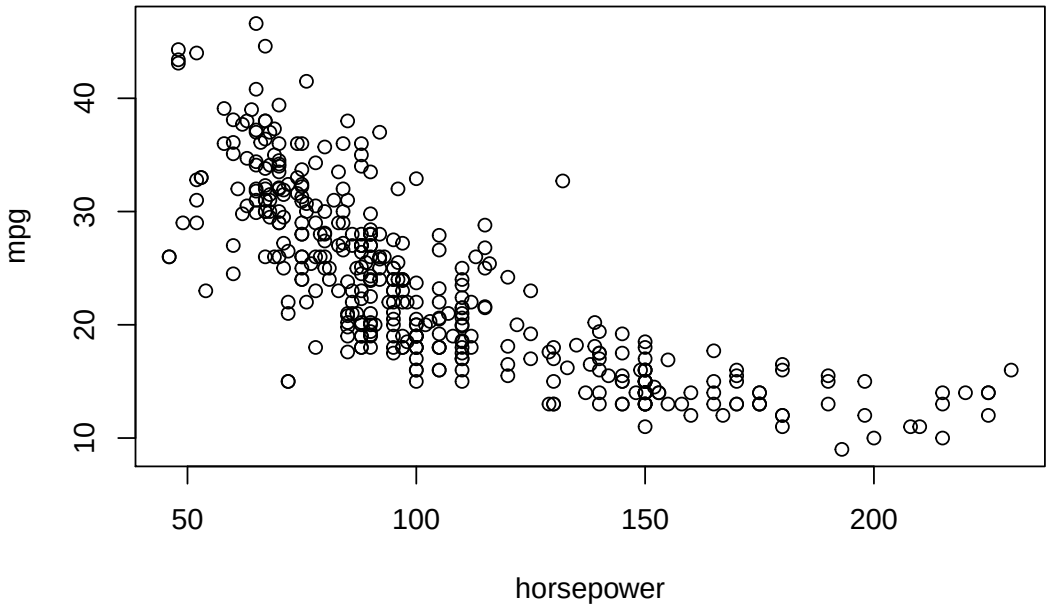

```
pairs(Auto)
```




```
pairs(~ mpg + displacement + horsepower + weight + acceleration, Auto)
```




```
plot(horsepower,mpg)
identify(horsepower,mpg,name)
```




```
## integer(0)
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
```

```
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##
##                  name
##  amc matador       :  5
##  ford pinto        :  5
##  toyota corolla    :  5
##  amc gremlin       :  4
##  amc hornet        :  4
##  chevrolet chevette:  4
##  (Other)           :365
```

```
summary(mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   17.00   22.75   23.45   29.00   46.60
```

2. Topic: Linear Models

The Candidate will understand key concepts concerning generalized linear models.

1. Describe and explain the components of, in particular, the exponential family of distributions and link functions.
2. Estimate parameters using least squares and maximum likelihood.
3. Interpret diagnostic tests of model fit and assumption checking, using both graphical and quantitative methods.
4. Select an appropriate model, considering
   - Distributions and link functions
   - Variable transformations and interactions
   - Pearson chi-square statistic t and F tests
   - AIC and BIC
   - Likelihood ratio test
5. Interpret model results with emphasis on using the model to answer the underlying business question.
6. Calculate and interpret predicted values, confidence, and prediction intervals.
7. Understand how approaches may differ compared to using an ordinary least squares model, including lasso, ridge regression, and KNN.

2.1 Basic Linear Regression ([REG] Ch. 2:1-8)

Define Pearson correlation.

The ordinary, or Pearson, correlation coefficient summarizes the strength of the relationship between two variables: $r = \frac{1}{(n-1)s_x s_y}\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})$, where the sample standard deviation $s_y = \sqrt{(n-1)^{-1}\sum_{i=1}^{n}(y_i-\overline{y})^2}$, with similar notation for s_x

Define unbiased estimates of the slope, residual standard deviation, and standard error of basic linear regression.

An estimator of of the variance of the residuals σ^2, the mean square error (MSE), is defined as $s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$. The slope estimate $b_1 = r\frac{s_y}{s_x} = \frac{1}{(n-1)s_x^2}\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})$. The standard error of b_1, the estimated standard deviation of b_1, is defined as $se(b_1) = \frac{s}{s_x\sqrt{n-1}}$.

2.2 Multiple Linear Regression I ([REG] Ch. 3:1-5)

List assumptions of the multiple linear regression models.

Observables Representation	Error Representation
F1. $E[y_i] = \beta_0 + \beta_1 x_{i1} + ... + \beta_k x_{ik}$.	E1. $y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_k x_{ik} + \epsilon_i$.
F2. $\{x_{i1}, ..., x_{ik}\}$ are non-stochastic variables.	E2. $\{x_{i1}, ..., x_{ik}\}$ are nonstochastic variables.
F3. $Var(y_i) = \sigma^2$.	E3. $E[\epsilon_i] = 0$ and $Var(\epsilon_i) = \sigma^2$.
F4. $\{y_i\}$ are independent random variables.	E4. $\{\epsilon_i\}$ are independent random variables.
F5. $\{y_i\}$ are normally distributed.	E5. $\{\epsilon_i\}$ are normally distributed.

List 3 properties of regression coefficient estimates.

1. Under assumptions F1-F4, the least squares regression estimator $b = (X'X)^{-1}X'y$ is an unbiased estimator of the parameter vector β;
2. and has variance $Var(b) = \sigma^2(X'X)^{-1}$.
3. Under assumption F1-F5, the least squares stimator is normally distributed.

Define residual standard deviation

With fitted values $\hat{y}_i = b_0 + b_1 x_{i1} + ... + b_k x_{ik}$. an estimator of σ^2, the mean square error (MSE), is defined as $s^2 = \frac{1}{n-(k+1)} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$. The positive square root, $s = \sqrt{s^2}$, is called the residual standard deviation.

Define coefficient of determination adjusted for degrees of freedom.

In the linear regression context, we may intepret total deviation (deviation without knowledge of the explanatory variables) to equal the deviation not explained by the explanatory variables plus deviation explained by the explanatory variables: $(y_i - \overline{y}) = (y_i - \hat{y}_i) + (\hat{y}_i - \overline{y}_i)$. Squaring each side and summing over all observations yields for the total sum of squared deviations $Total\ SS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{i=1}^{n} (\hat{y}_i - \overline{y}_i)^2 = Error\ SS + Regression\ SS$, where the sum of the cross-product terms turns out to be zero. A statistic that summarizes this relationship is the coefficient of determination, $R^2 = \frac{Regression\ SS}{Total\ SS}$. which can be interpreted to be the proportion of variability explained by the regression function.

Whenever an explanatory variable is added to the model, R^2 never decreases, whether or not the additional variable is useful. We would like a measure of fit that decreases when useless variables are entered into the model as explanatory variables. A widely used statistic is the coefficient of determination adjusted for degrees of freedom: $R_a^2 = 1 - \frac{(Error\ SS)/[n-(k+1)]}{(Total\ SS)(n-1)} = 1 - \frac{s^2}{s_y^2}$.

Define standard error, t-ratio, and confidence interval for regression coefficient estimates.

The standard error of b_j can be expressed as $se(b_j) = s\sqrt{(j+1)\text{stdiagonalelementof}(X'X)^{-1}}$. The t-ratio $t(b_j) = \frac{b_j}{se(b_j)}$ can be interpreted to be the number of standard errors that b_j is away from zero. In a t-test, the null hypothesis ($H_0 : \beta_j = 0$) is rejected in favor of the alternative if the absolute value of the t-ratio $|t(b_j)|$ exceeds a t-value, denoted $t_{n-(k+1),1-\frac{\alpha}{2}}$, equal to the $(1 - \frac{\alpha}{2})$th percentile from the t-distribution using $df = n-(k+1)$ degrees of freedom. Confidence intervals for parameters provide a range of reliability of a point estimate of the parameter: $b_j \pm t_{n-(k+1),1-\frac{\alpha}{2}}\ se(b_j)$.

Define partial correlation coefficient.

The partial correlation coefficient, which measures the correlation between y and the jth explanatory variable x_j, controlling for other explanatory variables, can be by running only one regression:

$r(y, x_j|x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_k) = \frac{t(b_j)}{\sqrt{t(b_j)^2+n-(k+1)}}$, where $t(b_j)$ is the t-ratio for b_j from a regression of y on $x_1, ..., x_k$ (including the variable x_j).

List 3 nonlinear functions of explanatory variables for linear regression

1. Binary Variables provide a numerical label for measurements of observations that fall in one of two distinct groups, or categories.
2. Transforming Explanatory Variables. Adding a pth order polynomial in x can be used to approximate general, unknown nonlinear functions of x.
3. Interaction Terms. If the marginal rate of increase of $E[y]$ per unit of x_1 differs for high values of x_2 when compared to low values of x_2, one way to represent this is to create a new interaction variable $x_3 = x_1 \times x_2$.

2.3 Variable Selection ([REG] Ch. 5:1-7)

Define forward selection, backward selection and stepwise regression.

Stepwise regression are procedures that employ t-tests to check the "significance" of explanatory variables entered into, or deleted from, the model.

- In the forward selection version, variables are added one at a time. In the first stage, out of all the candidate variables, the one that is most statistically significant is added to the model. At the next stage, with the first stage variable already included, the next most statistically significant variable is added. This procedure is repeated until all statistically significant variables have been added. Here, statistical significance is typically assessed using a variable's t -ratio – the cutoff for statistical significance is typically a predetermined t-value (e.g., two, corresponding to an approximate 95% significance level).
- The backward selection version works in a similar manner, except that all variables are included in the initial stage and then dropped one at a time (instead of added).
- More generally, an algorithm that adds and deletes variables at each stage is sometimes known as the stepwise regression algorithm:
    1. Consider all possible regressions using one explanatory variable. For each of the k regressions, compute the t -ratio for the slope. Choose that variable with the largest t-ratio. If the t-ratio does not exceed a prespecified t -value (e.g., two), then do not choose any variables and halt the procedure.
    2. Add a variable to the model from the previous step. The variable to enter is the one that makes the largest significant contribution. To determine the size of contribution, use the absolute value of the variable's t-ratio. To enter, the t-ratio must exceed a specified t-value in absolute value.
    3. Delete a variable to the model from the previous step. The variable to be removed is the one that makes the smallest contribution. To determine the size of contribution, use the absolute value of the variable's t-ratio. To be removed, the t-ratio must be less than a specified t-value in absolute value.
    4. Repeat the previous two steps until all possible additions and deletions are performed.

List drawbacks of stepwise regression.

1. The procedure "snoops" through a large number of models and may fit the data "too well."
2. There is no guarantee that the selected model is the best. The algorithm does not consider models that are based on nonlinear combinations of explanatory variables. It also ignores the presence of outliers and high leverage points.
3. In addition, the algorithm does not even search all 2^k possible linear regressions.
4. The algorithm uses one criterion, a t-ratio, and does not consider other criteria.
5. There is a sequence of significance tests involved. Thus, the significance level that determines the t-value is not meaningful.
6. By considering each variable separately, the algorithm does not take into account the joint effect of explanatory variables.
7. Purely automatic procedures may not take into account an investigator's special knowledge.

Define data snooping.

Data snooping occurs when the analyst fits a great number of models to a dataset. When explanatory variables are selected using the data, t-ratios and F-ratios will be too large, thus overstating the importance of variables in the model.

List 3 ways to standardize residuals for identifying outliers.

Using $e_i = y_i - \hat{y}_i$ as the ith residual, h_{ii} is the ith leverage of the explanatory variables, s is the residual standard deviatio, and $s_{(i)}$ is the residual standard deviation when running a regression after having deleted the ith observation, there are three commonly used definitions of standardized residual:

1. $\frac{e_i}{s}$, since sample standard deviation of the residuals is approximately s
2. $\frac{e_i}{s\sqrt{1-h_{ii}}}$, since the denominator is the estimated standard error for e_i.
3. $\frac{e_i}{s_{(i)}\sqrt{1-h_{ii}}}$, known as a studentized residual, omits the affect on the denominator from a large residual.

List 3 options for handling outliers

1. Include the observation in the usual summary statistics but comment on its effects. An outlier may be large but not so large as to skew the results of the entire analysis.
2. Delete the observation from the dataset. The observation may be deter mined to be unrepresentative of the population from which the sample is drawn.
3. Create a binary variable to indicate the presence of an outlier. This approach is similar to point deletion but allows the outlier to be formally included in the model formulation if special causes have been identified to explain such observations.

List 3 ways to analyze residuals after a preliminary model fit.

1. Calculate summary statistics and display the distribution of (standardized) residuals to identify outliers.
2. Calculate the correlation between the (standardized) residuals and additional explanatory variables to search for linear relationships.
3. Create scatter plots between the (standardized) residuals and additional explanatory variables to search for nonlinear relationships

Define leverage and Cook's distance.

An observation with an unusual explanatory variable a high leverage point. Large leverage values indicate that an observation may exhibit a disproportionate effect on the fit, essentially because it is distant from the other observations (when looking at the space of explanatory variables). The leverage for the ith observation is $h_{ii} = x_i'(X'X)^{-1}x_i$, which are calculated based on the explanatory variables X – values of the response variable are not used. A widely adopted convention is to declare an observation to be a high leverage point if the leverage exceeds three times the average, that is, if $h_{ii} > 3(k+1)/n$.

A measure that considers both the response and the explanatory variables is Cook's distance: $D_i = \frac{\sum_{j=1}^{n}(\hat{y}_j - \hat{y}_{j(i)})^2}{(k+1)s^2} = (\frac{e_i}{se(e_i)})^2 \frac{h_{ii}}{(k+1)(1-h_{ii})}$, where $\hat{y}_{j(i)}$ is the prediction of the jth observation. From the second equation, Cook's distance is composed of a measure for outliers times a measure for leverage.

List 4 options for handling high leverage points.

1. Include the observation in the summary statistics but comment on its effect.
2. Delete the observation from the datase if deemed not representative of some larger population.
3. Choose another variable to represent the information.
4. Use a nonlinear transformation of an explanatory variable

Define and list three facts about collinearity.

Collinearity, or multicollinearity, occurs when one explanatory variable is, or nearly is, a linear combination of the other explanatory variable.

1. Collinearity precludes us neither from getting good fits nor from making predictions of new observations.
2. Estimates of error variances and, therefore, tests of model adequacy, are still reliable.
3. In cases of serious collinearity, standard errors of individual regression coefficients are greater than in cases where, other things equal, serious collinearity does not exist. With large standard errors, individual regression coefficients may not be meaningful and it is difficult to detect the importance of a variable.

Define variance inflation factors.

$VIF_j = \frac{1}{1-R_j^2}$, $j = 1, 2, ..., k$, where R_j is the multiple correlation coefficient between x_j and linear combinations of the other x's from the regression usig x_j as the reponse and the other x's as the explanatory variables. Algebraicly, a larger VIF_j results in a larger standard error associated with the jth slope, b_j.

Define orthogonal variables and principal components.

Mathematically, two matrices are said to be orthogonal if $X_1'X_2' = 0$. Intuitively, each column of X_1 is uncorrelated with each column of X_2. Orthogonal variables can be created using the method of principal components. With this method, one uses a linear transformation of the matrix of explanatory variables of the form, $X^* = XP$, so that the resulting matrix X^* is composed of orthogonal columns.

List goodness of fit tests for linear models.

1. In linear regression, we can quantified this through the size of the typical error of residuals, include the coefficient of determination (R^2) and an adjusted version (R_a^2).
2. Akaike's information criterion for linear regression $AIC = n\ln(s^2) + n\ln(2\pi) + n + 3 + k$. Comparing models with the same number of variables (k) means that selecting a model with small values of AIC

leads to the same choice as selecting a model with small values of the residual standard deviation. Further, a small number of parameters means a small value of AIC , other things being equal.

3. Bayes information criterion (BIC), which gives a smaller weight to the penalty for complexity.
4. C_p statistic $C_p = \frac{((Error\ SS)_p}{s^2_{full}} - n + 2p$, where s^2_{full} is the mean square error from a regression on all available explanatory variables, and $(Error\ SS)_p$ is the error sum of squares using only $p-1$ explanatory variables (so that there are p regression coefficients).

Compare out-of-sample validation and leave-one-out cross-validation.

Out-of-sample validation splits the dataset into two subsamples. We call these the model development and validation subsamples, respectively. They are also known as training and testing samples, respectively. We initially develop one, or several, models on a first dataset. The models developed from the first set of data are called our candidate models. Then, the relative performance of the candidate models could be measured on a second set of data. In this way, the data used to validate the model is unaffected by the procedures used to formulate the model. But because the statistic is based on a random subset of the sample, its value will vary.

In leave-one-out cross-validation, the validation sample starts by consisting of a single observation and the development sample is based on the remainder of the dataset – the regression is computed and the squared error of the omitted observation is saved. This procedure is repeated, choosing the next one of the remaining observations to be left out of the development subset – the predicted residual sum of squares error (PRESS) is just the average over the n single squared error terms. In linear regression, a much easier computational formula can be derived: $PRESS = \sum_{i=1}^{n}(\frac{e_i}{1-h_{ii}})^2$, where e_i and h_{ii} represent the (i)th residual and everage from the regression fit using the complete dataset.

Define heteroscedasticity.

When the variability varies by observation, this is known as heteroscedasticity for "different scatter."

Describe how to test for heteroscedasticity.

1. Fit a regression model and calculate the model residuals, e_i.
2. Calculate squared standardized residuals, $e_i^{*2} = e_i^2/s^2$.
3. Fit a regression model of e_i^{*2} on a known vector of p variables z_i.
4. The test statistic is $LM = (Regress\ SS_z)/2$, where $Regress\ SS_z$ is the regression sum of squares from the model fit in the previous step.
5. Reject the null hypothesis if LM exceeds a percentile from a chi-square distribution with p degrees of freedom. The percentile is one minus the significance level of the test.

List 3 ways to handle heteroscedasticity.

1. We may define the empirical, or robust, estimate of the variance covariance matrix as $\hat{Var}(b) = (X'X)^{-1}(\sum_{i=1}^{n} e_i^2 x_i x_i')(X'X)^{-1}$ to adjust the corresponding "heteroscedasticity-consistent" standard errors.
2. Use a variation of least squares estimation by weighting observations. The idea is that, when minimizing the sum of squared errors using heteroscedastic data, the expected variability of some observations is smaller than others. Intuitively, it seems reasonable that the smaller the variability of the response, the more reliable that response and the greater weight that it should receive in the minimization procedure. Weighted least squares estimates can be expressed as $b_{WLS} = (X'WX)^{-1}X'Wy$, where

W is the diagonal matrix of known weights. For example, the population if the unit of analysis represents a geographical entity such as a state, or firm assets if the unit represents a firm.

3. Transform the dependent variable, typically with a logarithmic transformation, potentially altering a heteroscedastic dataset into a homoscedastic one. The transformation of the dependent variable affects both the skewness of the distribution and the heteroscedasticity.

2.4 Interpreting Regression Results ([REG] Ch. 6:1-3)

List arguments for and against Occam's razor

When selecting variables, analysts are often guided by the principle of parsimony, also known as Occam's razor, which states that when there are several possible explanations for a phenomenon, use the simplest one. There are several arguments for preferring simpler models:

- A simpler explanation is easier to interpret.
- Simple models, also known as parsimonious models, often do well on fitting out-of-sample data.
- Extraneous variables can cause problems of collinearity, leading to difficulty in interpreting individual coefficients

On the other hand, underfitting a model, by omitting important variables, may be a more serious error than including extraneous variables that add little to our ability to explain the data. Including extraneous variables decreases the degrees of freedom and increases the estimate of variability, which may be of less concern in some applications

List 4 major pitfalls in data collection.

1. Sampling Frame Error and Adverse Selection. Sampling frame error occurs when the sampling frame, the list from which the sample is drawn, is not an adequate approximation of the population of interest
2. Limited Sampling Regions. A limited sampling region can give rise to potential bias when we try to extrapolate outside of the sampling region.
3. Limited Dependent Variables, Censoring, and Truncation. In some applications, the dependent variable is constrained to fall within certain regions. This means that our assumption of normal errors is not strictly correct and may not even be a good approximation. Dependent variables can be restricted censored, with observed values no lower or higher than some limit. A more serious source of bias is when data are said to be truncated, or not recorded when depent variables of values below or above some threshold.
4. Omitted and Endogenous Variables. Sometimes we are prohibited from including all relevant variables. Omitted variables can lead to the presence of endogenous explanatory variables: An omitted variable can affect both the y and the x and in this sense induce a relationship between the two variables; then it is difficult to condition on the x when estimating a model for y.

2.5 Categorical Dependent Variables ([REG] Ch. 11:1-6)

List drawbacks of the linear probability model for binary dependent variables.

With binary dependent variable y, denote the probability that the response equals 1 by $\pi_i = \Pr(y_i = 1)$. The linear probability model takes the form, and hence predicts $E[y_i] = x_i^{'}\beta = \pi_i$

1. Fitted values can be poor. The expected response is a probability and thus must vary between 0 and 1. However, the linear combination $x_i^{'}\beta$ can vary between negative and positive infinity. This mismatch implies, for example, that fitted values may be unreasonable.
2. Heteroscedasticity. Linear models assume homoscedasticity (constant variance), yet the variance of the response depends on the mean that varies over observations.
3. Residual analysis is meaningless. The response must be eithe ra0or1, although the regression models typically regard distribution of the error term as continuous. This mismatch implies, for example, that the usual residual analysis in regression modeling is meaningless.

Compare the logit and probit link functions.

To circumvent the drawbacks of linear probability models, we consider alternative models in which we express the expectation of the response as a function of explanatory variables, $\pi_i = \pi(x_i\beta) = \Pr(y_i = 1|x_i)$. Two special cases are:

- logit: $\pi(x_i\beta) = \frac{1}{1+e^{-x_i\beta)}} = \frac{e^{x_i\beta}}{1+e^{x_i\beta}}$
- probit: $\pi(x_i\beta) = \Phi(x_i\beta)$, where Φ is the standard normal distribution function.

The inverse of the function, π^{-1} – called the link function – specifies that the form of the probability is linear in the explanatory variables, that is, $\pi^{-1}(\pi_i) = x_i\beta$.

Define the logit function, odds and log-odds in logistic regression.

Using $p = \pi(z) = (1 + e^{-z})^{-1}$, the inverse of π is calculated as $z = \pi^{-1}(p) = \ln(\frac{p}{1-p})$, which we define to be the logit function. With a logistic regression model, we represent the linear combination of explanatory variables as the logit of the success probability; that is, $x_i\beta = \text{logit}(\pi_\text{i})$. The transformation $\frac{p}{1-p}$ can be intepreted as the odds of success from a betting standpoint. The logit is the logarithmic odds function, also known as the log odds.

Define the log-likelihood function, maximum likelihood method, and score equations.

The customary method of estimation for logistic and probit models is maximum likelihood. The likelihood is the observed value of the probability function. The objective of maximum likelihood estimation is to find the parameter values that produce the largest likelihood. Finding the maximum of the logarithmic function yields the same solution as finding the maximum of the corresponding function. Because it is generally computationally simpler, we consider the logarithmic (or log-) likelihood. The log-likelihood is viewed as a function of the parameters, with the data held fixed. The method of maximum likelihood involves finding the values of parameters β that maximize the log-likelihood: $L(\beta) = \sum_i^n [y_i \ln \pi(x_i'\beta) + (1 - y_i) \ln(1 - \pi(x_i'\beta))]$.

The customary method of finding the maximum is taking partial derivatives with respect to the parameters of interest and finding roots of the resulting score equations: $\frac{\delta}{\delta\beta}L(\beta) = \sum_i^n x_i(y_i - \pi(x_i'\beta))\frac{\pi'(x_i'\beta)}{\pi(x_i'\beta)(1-\pi(x_i'\beta))} = 0$, where where π' is the derivative of π. The solution of these equations is the maximum likelihood estimator. For the logit function the score equations reduce to: $\frac{\delta}{\delta\beta}L(\beta) = \sum_i^n x_i(y_i - \pi(x_i'\beta)) = 0$, where $\pi(z) = 1/(1 + e^{-z})$.

Define the log-likelihood test

To test the overall model adequacy $H_0 : \beta = 0$, we use the statistic $LRT = 2 \times (L(b) - L_0)$, where L_0 is the maximized log-likelihood with only an intercept term. Under the null hypothesis, this statistic has a chi-square distribution with k (number of explanatory variables) degrees of freedom.

List models for nominal dependent variables.

When the dependent variable may take nominal values of unordered categories, standard maximum likelihood estimation is available with this framework: For an observation from the ith observation, denote the probability of choosing the jth category as $\pi_{ij} = \Pr(y_i = j)$, so that $\pi_{i1} + +\pi_{ic} = 1$. Thus the total log-likelihood is $L = \sum_i^n \sum_j^c y_{ij} \ln \pi_{ij}$. The remaining task is to specify an appropriate form for π.

- Generalized logit employs linear combinations of explanatory variables of the form $V_{ij} = x_i' \beta_j$, and probabilities $\Pr(y_i = j) = \pi_{ij} = \frac{e^{V_{ij}}}{\sum_{k=1}^c e^{V_{ik}}}$. So that probabilities sum to one, a convenient normalization is $\beta_c = 0$.
- Multinomial logit (also known as conditional logit) employs an alternative linear combinations of explanatory variables of the form $V_{ij} = x_{ij}' \beta$, where x_{ij} is a vector of explanatory variables thatThe total log-likelihood is $L = \sum_{i=1}^n \sum_{j=1}^c y_{ij} \ln \pi_{ij} = \sum_{i=1}^n [\sum_{j=1}^c y_{ij} x_{ij}' \beta - \ln(\sum_{j=1}^c exp(x_{ij}' \beta))]$.
- Nested logit is a hierarchical model, where inthe first stage, one chooses an alternative (say, the first) with probability $\pi_{i1} = \Pr(y_i = 1) = \frac{exp(V_{i1}}{exp(V_{i1} + [\sum_{k=2}^c exp(V_{ik}/\rho]^\rho}$. Then conditional on not choosing the first alternative, the probability of choose any one of the others follows a multinomial logit: $\frac{\pi_{ij}}{1-\pi_{i1}} = \Pr(y_i = j | y_i \neq 1) = \frac{exp(V_{i1}}{\sum_{k=2}^c exp(V_{ik}/\rho}$, $j = 1, ..., c$. The parameters ρ measures the association among the choices $j = 2, ..., c$. The value of $\rho = 1$ reduces to the multinomial logit model, where the odds ratio of any two alternatives is independent of the value of the other alternatives (called the independence of irrelevant alternatives, which can be a drawback in some cases)

List models for ordinal dependent variables.

When the response is an ordered categorical variable (also known as an ordinal dependent variable), models are based on cumulative probabilities of the form $\Pr(y \leq j) = \pi_1 + ... + \pi_j$, $j = 1, ..., c$.

- Cumulative logits uses $\text{logit}(Pr(y \leq j)) = \ln(\frac{Pr(y \leq j)}{1 - Pr(y \leq j)} = \ln(\frac{\pi_1 + ... + \pi_j}{\pi_{j+1} + ... + \pi_c})$. The simplest cumulative logit model does not use any explanatory variables in the cut-point parameters: $\text{logit}(Pr(y \leq j)) = \alpha_j$, $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_c$. The proportional odds model incorporates explanatory variables: $\text{logit}(Pr(y \leq j)) = \alpha_j + x_i' \beta$
- Cumulative probit assumes that $y_i^* - x_i \beta$ has a standard normal distribution. Then, $Pr(y_i \leq j) = Pr(y_i^* \leq \alpha_j) = \Phi(\alpha_j - x_i' \beta)$.

2.6 Count Dependent Variables ([REG] Ch. 12:1-4)

Describe poisson regression models for count dependent variables.

Count dependent variables take on values of zero, one, and two, and so on, which describes a number of events. The Poisson is a fundamental distribution used for counts that has probability mass function $Pr(y = j) = \frac{\mu^j}{j!} e^{-\mu}$, $j = 0, 1, 2, ...$ The parameter μ can be interpreted to be the mean of the distribution, and one can show that the mean equals the variance for this distribution. The log-likehood is $\ln(\mu^{y_i} e^{-\mu} / y!)$, with maximum at the average counts $\hat{\mu} = \overline{y}$ and estimated probabilities $\hat{p}_j = \frac{\hat{\mu}^j}{j!} e^{-\mu}$. To compare observed observed and fitted counts, a widely used goodness of fit statistic is Pearson's chi-square statistic, given by

$\sum_j \frac{n_j - n\hat{p}_j}{n\hat{p}_j}$. Under the null hypothesis that the Poisson distribution is a correct model, this statistic has a large sample chi-square distribution where the degrees of freedom is the number of categories minus one minus the number of estimated parameters. More generally, we wish to allow the mean to vary according to information contained in other explanatory variables: $E[y_i] = \mu_i = exp(x_i'\beta)$, which corresponds to a logarithmic link function $\ln \mu_i = x_i\beta$.

Describe the flexibility of negative binomial, zero-inflation, hudle, heterogeneity and latent class models.

- Negative binomial is a widely used model for counts. The requirement of the Poisson that the mean equal the variance, known as equidispersion, is not satisfied for many datasets of interest – if the variance exceeds the mean, then the data are said to be overdispersed; a less common case occurs when the variance is less than the mean, known as underdispersion. wo parameters describing the negative binomial distribution, it has greater flexibility for fitting data, With two parameters r and p describing the negative binomial distribution, it has greater flexibility for fitting data: $Pr(y = j) = \binom{j+r-1}{r-1} p^r (1-p)^j$. To help interpret the parameters of the model, straightforward calculations show that $E[y] = r(1-p)/p$ and $Var[y] = r(1-p)/p^2$. It can be shown that the Poisson is a limiting case of the negative binomial, and that negative binomial distribution arises from a mixture of the Poisson variables.

- A zero-inflated model represents the dependent variables as a mixture of a point mass at zero and another claims frequency distribution, say Poisson or negative binomial, to account for an 'excess" number of zeros relative to the specified distribution.

- A hurdle model provides another mechanism to modify basic count distributions to represent situations with an excess number of zeros. Hurdle models can be motivated by sequential decision-making processes, where one needs to pass the first "hurdle" decision to address the second amount. A logit model might be suitable for representing the probability used for the first decision, and a count distribution like the Poisson for the second.

- In a heterogeneity model, one allows one or more model parameters to vary randomly. The motivation is that the random parameters capture unobserved features of a subject. For example, suppose that α_i represents a random parameter and that y_i given α_i has conditional mean $exp(\alpha_i + x_i'\beta)$. We interpret α_i, called a heterogeneity component, to represent unobserved subject characteristics that contribute linearly to the systematic component $x_i\beta$. Two common distributions for the distribution of α_i are the log-gamma and the lognormal.

- A latent class model employs an underlying classification of the dataset that is unobserved, or latent. We use mixture models to modify basic count distributions but now assume that the mixture is a discrete random variable that we interpret to be the latent class. This model is intuitively pleasing in that it corresponds to an analyst's perception of the behavior of the world, and flexible in the sense that the model readily accommodates under- and overdispersion, long tails, and bimodal distributions. However, this flexibility also leads to difficulty regarding computational issues. There is a possibility of multiple local maxima when estimating via maximum likelihood. Convergence can be slow compared to other methods.

2.7 Generalized Linear Models ([REG] Ch. 13:1-6)

List 3 common features of GLM models.

1. We can express the mean response as a function of linear combinations of explanatory variables. In the GLM context, it is customary to use $\mu_i = E[y_i]$ for the mean response, call $\eta_i = x_i\beta$ the systematic component of the model, and relate the systematic component to the mean through the expression $\eta_i = x_i\beta = g(\mu_i)$ where $g(\cdot)$ is known as the link function. For example:
   - $x_i\beta = \mu_i$, for (normal) linear regression
   - $x_i\beta = exp(\mu_i)/(1 + exp(\mu_i))$, for logistic regression
   - $x_i\beta = \ln(\mu_i)$, for Poisson regression
2. The distribution of the dependent variables, which we often draw from the linear exponential family of distributions, an extension of the exponential distribution, which includes the normal, Bernoulli, and Poisson distributions as special cases.
3. The robustness of inference to the choice of distributions: responses need not be normally distributed for statistical inference procedures to be effective.

Define linear exponential family of distributions.

The distribution of the linear exponential family is $f(y;\theta,\phi) = exp(\frac{y\theta - b(\theta)}{\phi} + S(y,\phi))$. Here, y is a dependent variable which may be discrete, continuous or a mixture; and θ is the parameter of interest. The quantity ϕ is a scale parameter. The term $b(\theta)$ depends only on the parameter θ, not on the dependent variable. The statistic $S(y,\phi)$ is a function of the dependent variable and the scale parameter, not the parameter θ. some straightforward calculations show that $E[y] = b'(\theta)$ and $Var[y] = \phi b''(\theta)$.

Define the link function and canonical link.

In a regression model, the link function serves to relate the mean $E[y_i] = \mu_i$ with the systematic component and thus to the model parameters. Combinations of explanatory variables, including the linear case $x_i'\beta$, may vary between negative and positive infinity, but Poisson means vary between zero and infinity. The link function maps the domain of the mean function onto the whole real line. There may be several link functions that are suitable for a particular distribution. The choice of g that is the inverse of $b'(\theta)$ is called the canonical link. With this choice, the systematic component equals the parameter of interest: recall that $\eta = g(\mu)$ and $\mu = b'(\theta)$; hence if $g^{-1} = b'$ then $\eta = g(b'(\theta)) = \theta$.

Mean Functions and Canonical Links for Selected Distributions:

Distribution	Mean function $b(\theta)$	Canonical link $g(\mu)$
Normal	θ	μ
Bernoulli	$e^\theta/(1+e^\theta)$	$\text{logit}(\mu)$
Poisson	e^θ	$\ln \mu$
Gamma	$-1/\theta$	$-1/\mu$
Inverse Gaussian	$(-2\theta)^{\frac{1}{2}}$	$-1/(2\mu^2)$

Explain why R^2 is not useful for nonlinear models.

R^2 is not a useful statistic in nonlinear models, in part because of the analysis of variance decomposition is no longer valid. The most widely cited goodness-of-fit statistic in linear regression models, R^2 is based on the decomposition: $\sum_i (y_i - \overline{y})^2 = \sum_i (y_i - \hat{y}_i)^2 + \sum_i (\hat{y}_i - \overline{y})^2 + 2 \times \sum_i (y_i - \hat{y}_i)(\hat{y}_i - \overline{y})$, or in words: *Total SS = Error SS + Regression SS + 2 × Sum of Cross-Products*. The difficulty with nonlinear models is that the *Sum of Cross-Products* term rarely equals zero. Thus, one gets different statistics when defining R^2 as *(Regression SS/Total SS)* as compared to *(1 – Error SS/Total SS)*.

Define the Pearson chi-square, AIC, BIC and deviance goodness-of-fit tests

A widely cited goodness-of-fit measure is the Pearson chi-square statistic. In the GLM context, we suppose that $\hat{\mu}_i$ is an estimator of the mean function $E[y_i] = \mu_i$, hence $\phi\nu(\hat{\mu}_i)$ is an estimator of the variance function $Var(y_i)$. Then, the Pearson chi-square statistic is defined as $\sum_i (y_i - \hat{\mu}_i)^2/(\phi\nu(\hat{\mu}_i))$. If the model is specified correctly, then this statistic should be approximately $n - (k+1)$.

Maximizing the likelihood does not impose sufficient structure on the problem because we know that we can always make the likelihood greater by introducing additional parameters. A reasonable alternative is to minimize Akaike's information criterion $AIC = -2 \times L(\theta_{MLE}) + 2kn$, where the additional term on k, number of parameters, is a penalty for the complexity of the model.

Bayesian information criterion, defined as $BIC = -2 \times L(\theta_{MLE}) + k \ln n$ where n is the number of observations, gives greater weight to the number of parameters, k. Other things being equal, BIC will suggest a more parsimonious model than AIC.

A goodness-of-fit measure that is specific to GLM modeling is the deviance statistic. We start with the notion a saturated model, where there are as many parameters as observations, θ_i, $i = 1, ..., n$. A saturated model provides the best possible fit, which yields the parameter estimate $\theta_{i,SAT}$ as the solution of $y_i = b'(\theta_{i,SAT})$ which has the largest possible value of the log-likelihood, denoted $L(\theta_{SAT}$. Then, for a generic estimator θ, the scaled deviance statistic is defined as $D^*(\hat{\theta}) = 2 \times L(\theta_{SAT}) - L(\theta)$. In linear exponential families, one multiplies by the scaling factor ϕ to define the deviance statistic $D(\hat{\theta}) = \phi D^*(\hat{\theta})$. The deviance statistic reduces to the following forms for three special cases:

- Normal: $D(\hat{\mu}) = \sum_i (y_i - \hat{\mu}_i)^2$
- Bernoulli: $D(\hat{\pi}) = \sum_i \{y_i \ln \frac{y_i}{\pi_i} + (1 - y_i) \ln \frac{1-y_i}{1-\hat{\pi}_i}\}$
- Poisson: $D(\hat{\mu}) = \sum_i \{y_i \ln \frac{y_i}{\mu_i} + (y_i - \hat{\mu}_i)\}$

2.8 Simple Linear Regression ([ISL] Ch. 3.1)

Define residual and RSS.

$e_i = y_i - \hat{y}_i$ represents the ith residual – this is the difference between the ith observed response value and the ith response value that is predicted by our linear model. We define the residual sum of squares as $RSS = e_1^2 + e_2^2 + ... + e_n^2$ or equivalently as $RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 ... (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$

Recognize and apply the least squares coefficient estimates.

$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^n (x_i - \overline{x})^2}$, $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$

Interpret the least squares coefficients.

The least squares approach chooses the least squares coefficients β_0 andβ_1 to minimize the RSS.

Define population regression line and least squares line.

The model given by $Y = \beta_0 + \beta_1 X + \epsilon$ defines the population regression line, which is the best linear approximation to the true relationship between X and Y. The least squares regression coefficient estimates that minimizedd the RSS characterize the least squares line $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$

Define the concept of bias and unbiased estimators.

Do we expect the estimate to equal the true population parameter? An unbiased estimator does not systematically over- or under-estimate the true parameter. If we could average a huge number of estimates

from a huge number of sets of observations, then this estimate would be spot on.

Define standard error and residual standard error.
Standard error tells us the average amount that the estimate differs from the actual value. The residual standard error is the estimate of the (square root of the) variance of the residuals $\hat{\sigma} \equiv RSE = \sqrt{RSS/(n-2)}$.

Calculate standard error of a statistic.
The standard errors associated with linear regression coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are:
$\text{SE}(\hat{\beta}_0^2) = \sigma^2[\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^n (x_i - \overline{x})^2}]$, $\text{SE}(\hat{\beta}_1^2) = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \overline{x})^2}$

Calculate the 95% confidence interval.
For linear regression, the 95% confidence interval for β approximately takes the form $\hat{\beta} \pm 2 \cdot \text{SE}(\hat{\beta})$

Describe null and alternative hypothesis.
The most common hypothesis test involves testing the null hypothesis of H_0: There is no relationship between X and Y; versus the alternative hypothesis H_a : There is some relationship between X and Y.

Calculate the t-statistic.
$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta})}$

Describe how accuracy of a linear regression can be assessed.
The quality of a linear regression fit is typically assessed using two related quantities: the residual standard error (RSE) and the R^2 statistic. The RSE provides an absolute measure of lack of fit of the model The R^2 statistic provides an alternative measure of fit. It takes the form of a proportion – the proportion of variance explained – and so it always takes on a value between 0 and 1, and is independent of the scale.

Recognize and apply R^2 statistic.
$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$, where TSS, the total sum of squares, is the amount of variability inherent in the response before the regression is performed. An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression. A number near 0 indicates that the regression did not explain much of the variability in the response.

Define total sum of squares.
$\text{TSS} = \sum(y_i - \overline{y})^2$ measures the total variance in the response Y.

2.9 Multiple linear regression ([ISL] Ch. 3.2)

Interpret the coefficients of a multiple linear regression.
The coefficient β_j quantifies the association between the jth predictor and the response. It is the average effect on Y of a one unit increase in X_j, holding all other predictors fixed.

Describe how relationship between response and predictors is tested in a multiple linear regression.
In the multiple regression setting with p predictors, we need to ask whether all of the regression coefficients are zero. We use a hypothesis test to answer this question. We test the null hypothesis, $H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$, versus the alternative H_a : at least one β_j is non-zero. This hypothesis test is performed by computing the F-statistic. Sometimes, we want to test that a particular subset of q of the coefficients

are zero. In this case we fit a second model that uses all the variables except those last q, then compute residual sum of squares for that model and the appropriate F-statistic.

Recognize and apply the F-statistic.

Tests whether all of the regression coefficients are zero: $F = \frac{(TSS-RSS)/p}{RSS/(n-p-1)}$

Describe how important variables can be decided in multiple regression.

This task is referred to as variable selection. Ideally, we can try out a lot of different models, each containing a different subset of the predictors, then se- lect the best model out of all of the models that we have considered. Various statistics can be used to judge the quality of a model, such as Mallow's C_p, Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R^2.

Define forward selection, backward selection, and mixed selection.

Trying out every possible subset of the predictors may be infeasible. Three classical automated and efficient approaches to choose a smaller set of models to consider are:

- Forward selection: We begin with the null model – a model that contains an intercept but no predictors. We then fit p simple linear regressions and add to the null model the variable that results in the lowest RSS. We then add to that model the variable that results in the lowest RSS for the new two-variable model,. This approach is continued until some stopping rule is satisfied.
- Backward selection: We start with all variables in the model, and remove the variable with the largest p-value – that is, the variable that is the least statistically significant. The new (p-1)-variable model is fit, and the variable with the largest p-value is removed. This procedure continues until a stopping rule is reached.
- Mixed selection: This is a combination of forward and backward selection. We start with no variables in the model, and as with forward selection, we add the variable that provides the best fit. We continue to add variables one-by-one. If at any point the p-value for one of the variables in the model rises above a certain threshold, then we remove that variable from the model. We continue to perform these forward and backward steps until all variables in the model have a sufficiently low p-value, and all variables outside the model would have a large p-value if added to the model.

Describe the tools used to examine model fit for multiple regression.

Two of the most common numerical measures of multiple regression model fit are the RSE and R^2, the fraction of variance explained. These quantities are computed and interpreted in the same fashion as for simple linear regression.

2.10 Considerations in the regression model ([ISL] Ch. 3.3)

Define dummy variables.

If a qualitative predictor (also known as a factor) only has two levels, or possible values, then incorporating it into a regression model is very simple. We simply create an indicator or dummy variable that takes on two possible numerical values: 0 or 1.

Describe how qualitative variables with more than two levels can be used in multiple regression.

When a qualitative predictor has more than two levels, a single dummy variable cannot represent all possible values. In this situation, we can create additional dummy variables. There will always be one

fewer dummy variable than the number of levels. The level with no dummy variable is known as the baseline. The estimated coefficients (of the dummy variables) can be intepreted as the difference in the average response between that level and the baseline category.

Describe additive and linear assumptions for linear regression model.

Two of the most important assumptions of the standard linear regression model state that the relationship between the predictors and response are additive and linear. The additive assumption means that the effect of changes in a predictor on the response is independent of the values of the other predictors. The linear assumption states that the change in the response due to a one-unit change in a predictor is constant, regardless of the value of the predictor.

Define interaction effect.

Suppose the assumption that the effect on the response of one predictor is independent of the value of another predictor is incorrect. One way of extending this model to allow for interaction effects is to include a third predictor, called an interaction term, which is constructed by computing the product of the values of the two variables.

Interpret the coefficients of an interaction term.

We can interpret coefficient as the increase in the effectiveness of one predictor for a one unit increase in the other predictor (or vice-versa).

Describe hierarchical principle for multiple regression.

If we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant.

Define polynomial regression.

A very simple way to directly extend the linear model to accommodate non-linear relationships, using polynomial regression, is to include transformed versions of the predictors in the model. We can add a quadratic term or several polynomial functions of the predictors in the regression model, and use standard linear regression software to estimate coefficients in order to produce a non-linear fit.

Describe the potential problems, such as non-linearity, correlation of error terms, non-constant variance, outliers, high-leverage points, and collinearlity, for linear regression model.

1. **Non-linearity of the Data**: If the true relationship is far from linear, then virtually all of the conclusions that we draw from the fit are suspect. In addition, the prediction accuracy of the model can be significantly reduced.

2. **Correlation of Error Terms**: If in fact there is correlation among the error terms, then the estimated standard errors will tend to underestimate the true standard errors. As a result, confidence and prediction intervals will be narrower than they should be.

3. **Non-constant Variance of Error Terms**: The standard errors, confidence intervals, and hypothesis tests associated with the linear model rely upon the assumption that the error terms have a constant variance. Unfortunately, it is often the case that the variances of the error terms are non-constant. For instance, the variances of the error terms may increase with the value of the response.

4. **Outliers**: An outlier is a point for which its response value is far from the value predicted by the model. Outliers can arise for a variety of reasons, such as incorrect recording of an observation during data collection. Removing the outlier may (or may not) have little effect on the least squares line.

But can cause other problems such as the RSE, which is used to compute all confidence intervals and can have implications for the interpretation of the fit.

5. **High Leverage Points**: Observations with high leverage have an unusual value for a predictor value. Removing the high leverage observation may have substantial impact on the estimated least squares line. It is cause for concern if the least squares line is heavily affected by just a couple of observations, because any problems with these points may invalidate the entire fit.

6. **Collinearity**: This refers to the situation in which two or more predictor variables are closely related to one another. The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variables on the response. It reduces the accuracy of the estimates of the regression coefficients, and causes the standard errors to grow. As a result, in the presence of collinearity, we may fail to reject null hypothesis that the coefficients are zero. This means that the power of the hypothesis test – the probability of correctly detecting a non-zero coefficient – is reduced.

Define heteroscedasticity.
Non-constant variances in the error terms of a regression model.

Define power of a hypothesis test.
The probability of a hypothesis test correctly detecting a non-zero coefficient.

Define multicollinearity and variance inflation factor.
Multicollinearity is the situation when collinearity exists between three or more variables even if no pair of variables has a particularly high correlation. A better way to assess multicollinearity, than inspecting the correlation matrix of predictor variables, is to compute the variance inflation factor: $VIF(\hat{\hat{\beta}}_j) = 1/(1 - R^2_{X_j|X_{-j}})$. where $R^2_{X_j|X_{-j}}$ is the R^2 from a regression of X_j onto all of the other predictors. If $R^2_{X_j|X_{-j}}$ is close to one, then collinearity is present, and so the VIF will be large. The smallest possible value for VIF is 1, which indicates the complete absence of collinearity. As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity.

2.11 Comparison of Linear Regression with K-Nearest Neighbors ([ISL] Ch. 3.5)

Describe KNN regression
The KNN regression method is closely related to the KNN classifier. Given a value for K and a prediction point x_0, KNN regression first identifies the K training observations that are closest to x_0, represented by N_0. It then estimates $\hat{f}(x_0)$ using the average of all the training responses in N_0.

2.12 Lab: Linear Regression ([ISL] Ch 3.6)


```r
library(MASS)
library(ISLR)

# Simple Linear Regression

#fix(Boston)
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```r
#lm.fit=lm(medv~lstat)
lm.fit=lm(medv~lstat,data=Boston)
attach(Boston)
lm.fit=lm(medv~lstat)
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```r
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
coef(lm.fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

```
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

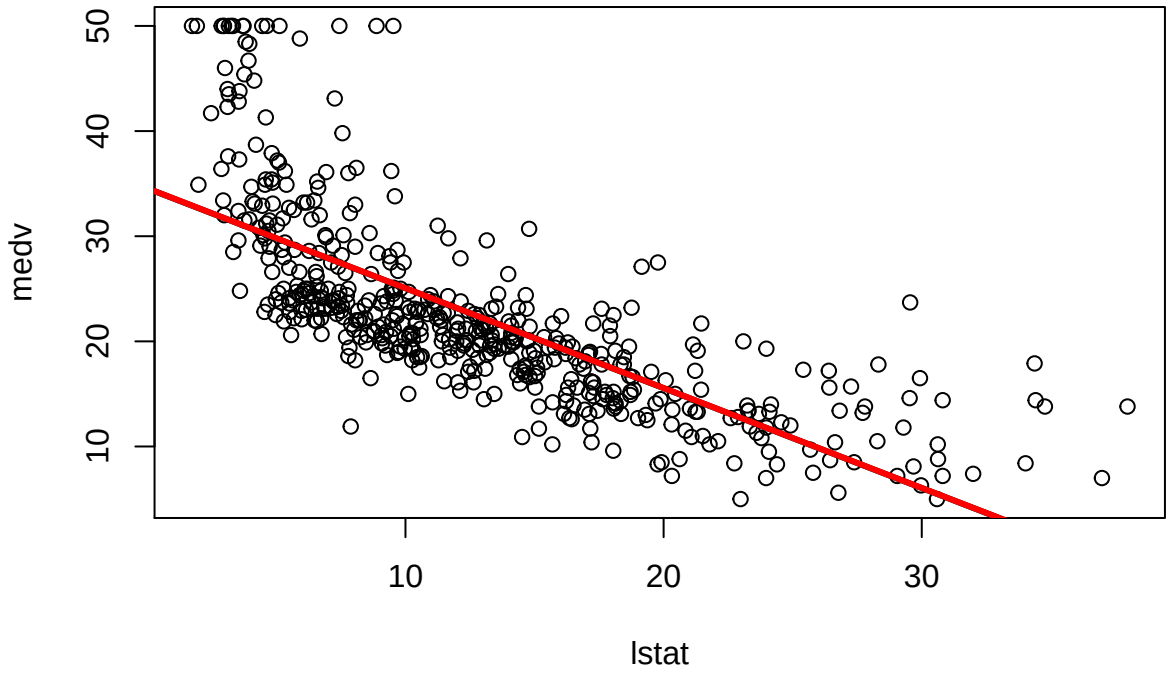
```
predict(lm.fit,data.frame(lstat=(c(5,10,15))), interval="confidence")
```

```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

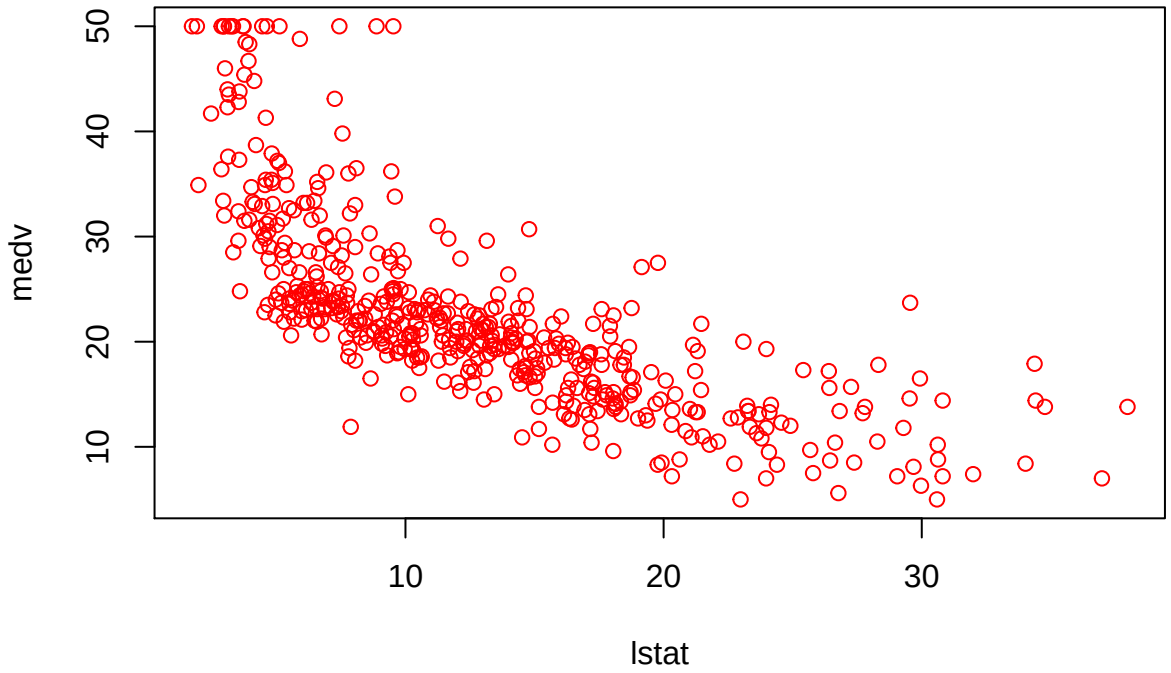
```
predict(lm.fit,data.frame(lstat=(c(5,10,15))), interval="prediction")
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

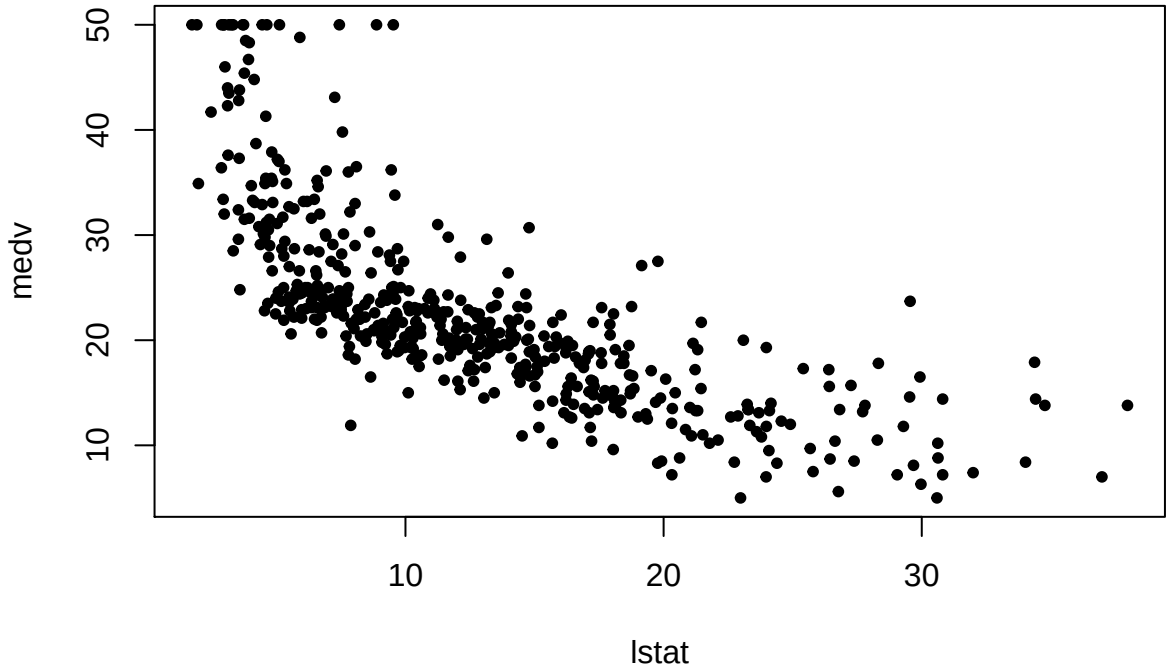
```
plot(lstat,medv)
abline(lm.fit)
abline(lm.fit,lwd=3)
abline(lm.fit,lwd=3,col="red")
```



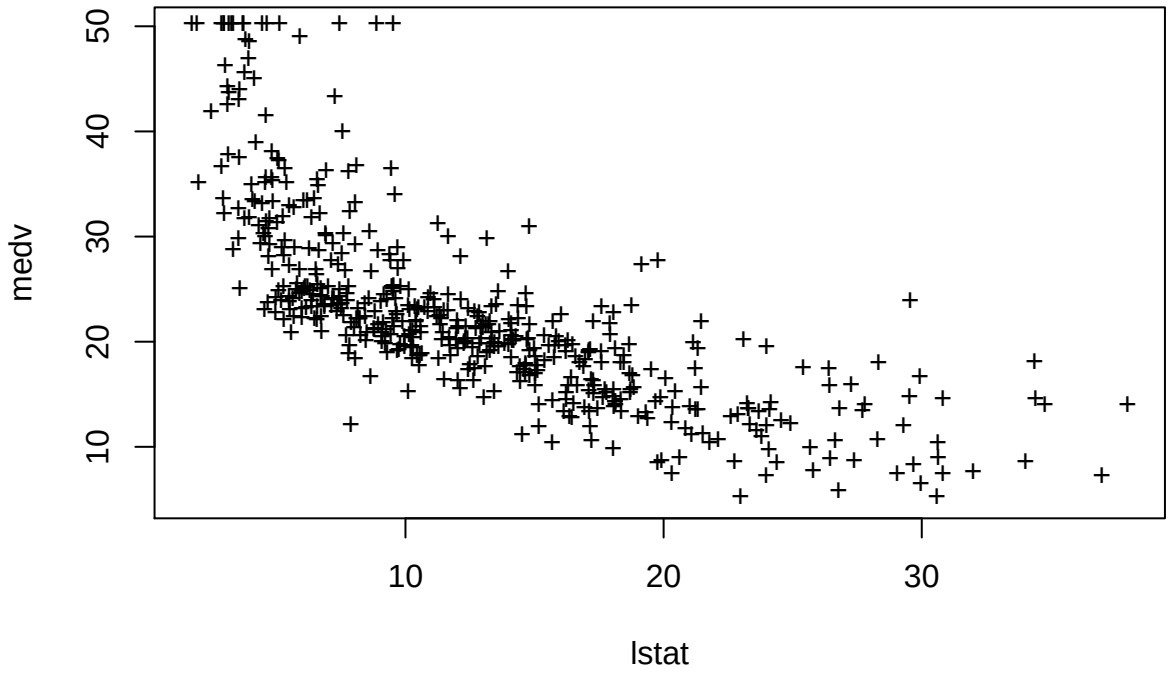

```
plot(lstat,medv,col="red")
```



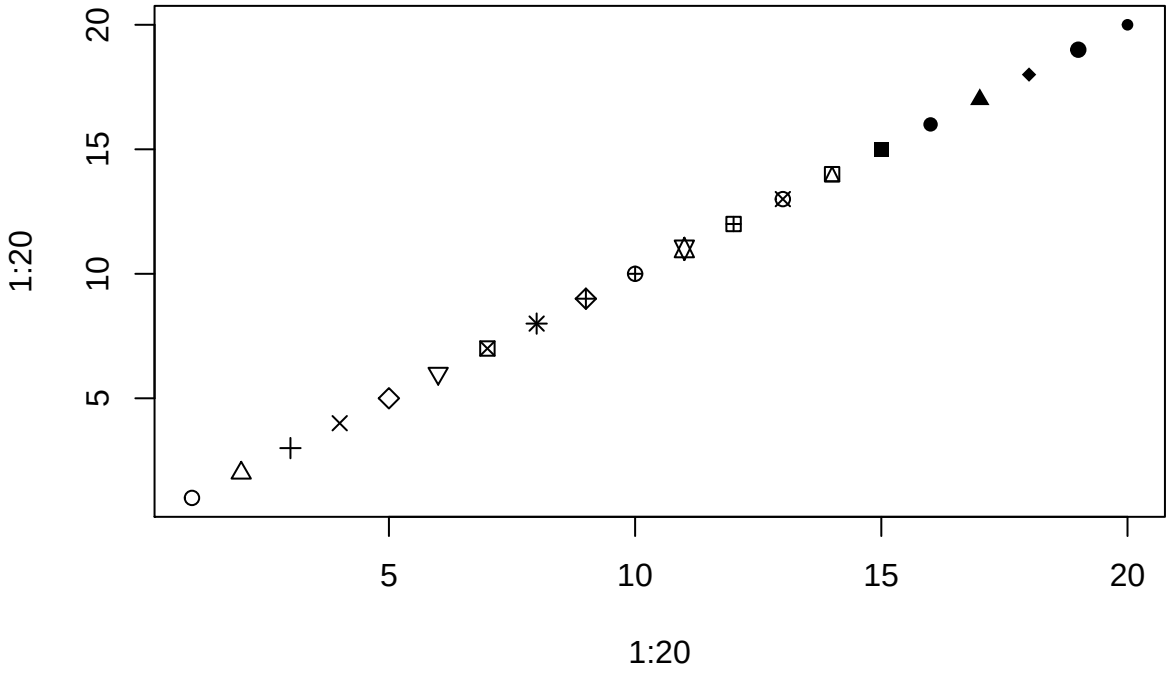

```
plot(lstat,medv,pch=20)
```



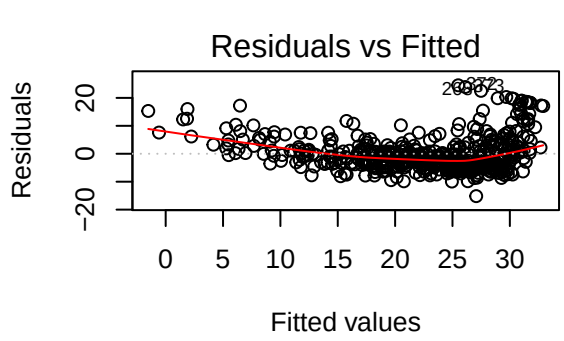

```
plot(lstat,medv,pch="+")
```

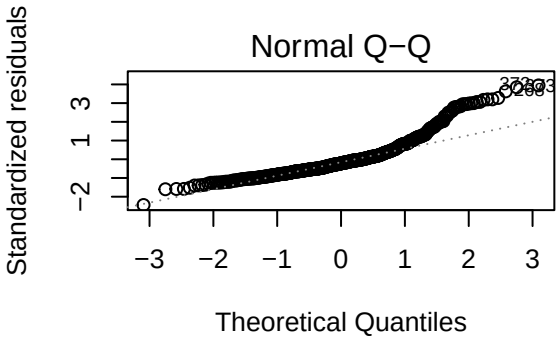


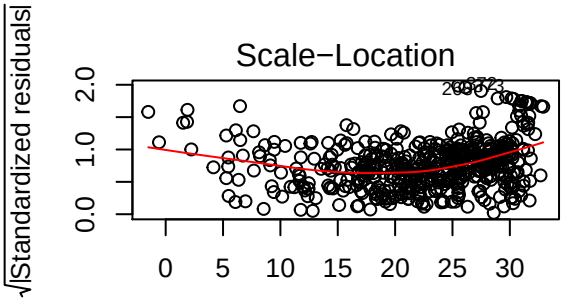

```
plot(1:20,1:20,pch=1:20)
```

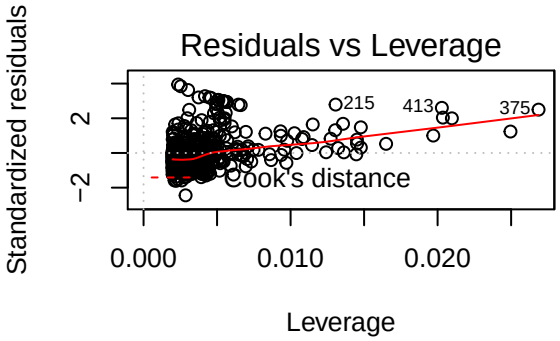



```
par(mfrow=c(2,2))
plot(lm.fit)
```










```
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
plot(hatvalues(lm.fit))
which.max(hatvalues(lm.fit))
```

```
## 375
## 375
```

```
# Multiple Linear Regression

lm.fit=lm(medv~lstat+age,data=Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
```

```
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
lm.fit=lm(medv~.,data=Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
library(car)
```

```
## Loading required package: carData
```

```
vif(lm.fit)
```

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.792192 2.298758 3.991596 1.073995 4.393720 1.933744 3.100826 3.955945
##      rad      tax  ptratio    black    lstat
## 7.484496 9.008554 1.799084 1.348521 2.941491
```

```
lm.fit1=lm(medv~.-age,data=Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
lm.fit1=update(lm.fit, ~.-age)

# Interaction Terms

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

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

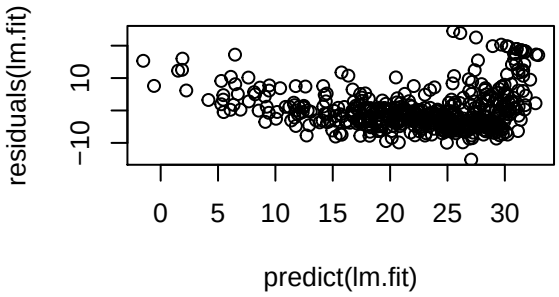
```
# Non-linear Transformations of the Predictors

lm.fit2=lm(medv~lstat+I(lstat^2))
summary(lm.fit2)
```

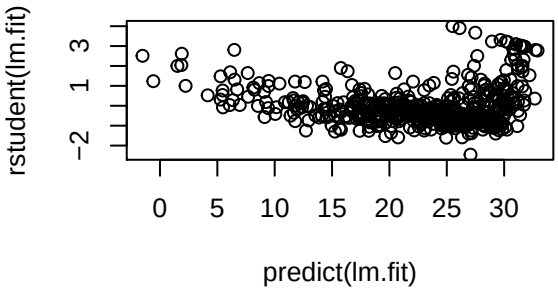
```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

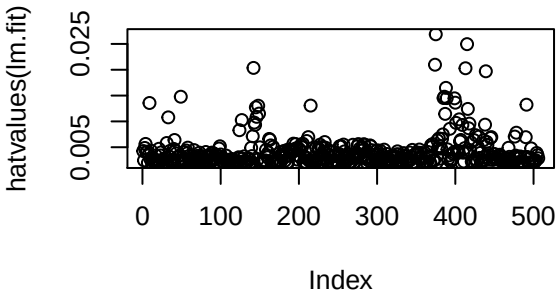
```
lm.fit=lm(medv~lstat)
anova(lm.fit,lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

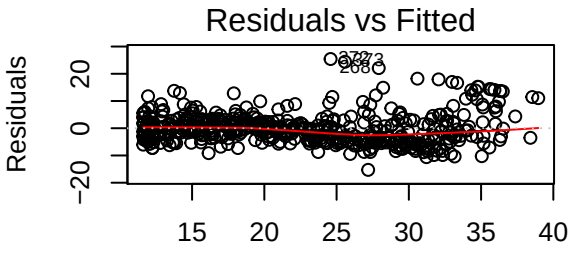
```
par(mfrow=c(2,2))
```

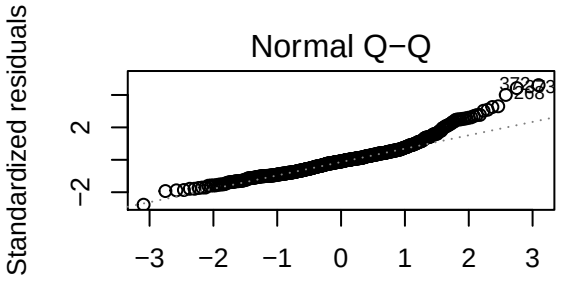


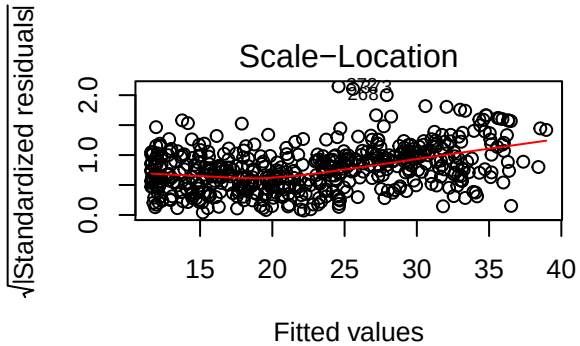


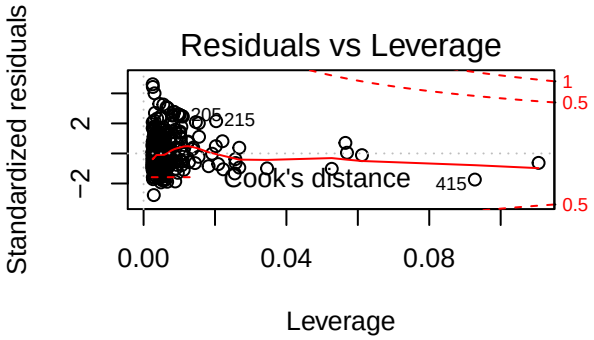



```
plot(lm.fit2)
```










```
lm.fit5=lm(medv~poly(lstat,5))
summary(lm.fit5)
```

```
##
## Call:
```

```
## lm(formula = medv ~ poly(lstat, 5))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

```
summary(lm(medv~log(rm),data=Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
# Qualitative Predictors

#fix(Carseats)
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

```
lm.fit=lm(Sales~.+Income:Advertising+Price:Age,data=Carseats)
summary(lm.fit)
```

```
##
## Call:
```

```
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9208 -0.7503  0.0177  0.6754  3.3413
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         6.5755654  1.0087470   6.519 2.22e-10 ***
## CompPrice           0.0929371  0.0041183  22.567  < 2e-16 ***
## Income              0.0108940  0.0026044   4.183 3.57e-05 ***
## Advertising         0.0702462  0.0226091   3.107 0.002030 **
## Population          0.0001592  0.0003679   0.433 0.665330
## Price              -0.1008064  0.0074399 -13.549  < 2e-16 ***
## ShelveLocGood       4.8486762  0.1528378  31.724  < 2e-16 ***
## ShelveLocMedium     1.9532620  0.1257682  15.531  < 2e-16 ***
## Age                -0.0579466  0.0159506  -3.633 0.000318 ***
## Education          -0.0208525  0.0196131  -1.063 0.288361
## UrbanYes            0.1401597  0.1124019   1.247 0.213171
## USYes              -0.1575571  0.1489234  -1.058 0.290729
## Income:Advertising  0.0007510  0.0002784   2.698 0.007290 **
## Price:Age           0.0001068  0.0001333   0.801 0.423812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 386 degrees of freedom
## Multiple R-squared:  0.8761, Adjusted R-squared:  0.8719
## F-statistic:   210 on 13 and 386 DF,  p-value: < 2.2e-16
```

```
attach(Carseats)
contrasts(ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

```
# Writing Functions

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

```
## [1] "The libraries have been loaded."
```

2.13 Logistic regression ([ISL] Ch. 4.1-3)

Describe the limitations of linear regression for categorical response variable.
In general there is no natural way to convert a qualitative response variable with more than two levels into a quantitative response that is ready for linear regression.

Recognize and apply the logistic function, odds, and log-odds.
The logistic function gives outputs between 0 and 1 for all input values hence can represent probabilities: $p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$. The quantity $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$ is called the odds, and can take on any value between 0 and ∞. Values of the odds close to 0 and ∞ indicate very low and very high probabilities of default, respectively. Taking the logarithms, the left-hand side is called the log-odds or logit: $\log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X$.

Recognize and apply the likelihood function.
The likelihood function $l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} p(x_j)$. is maximized to choose the estimates $\hat{\beta}_0, \hat{\beta}_1$. The basic intuition behind using maximum likelihood to fit a logistic regression model is to seek coefficient estimates such that the predicted probability $\hat{p}(x_i)$ for each data point corresponds as closely as possible to its observed class.

Define odds and log odds
Odds expresses a probability value divided by 1 minus the value. Log odds is the logarithm of the odds value.

Interpret the logistic regression coefficients with single regressor.
Increase in the log odds of a one-unit increase in the regressor.

Recognize and apply predictions using the logistic regression.
Once the coefficients have been estimated, the predicted probability is $\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\beta_0+\beta_1 X}}$.

Interpret the multiple logistic regression coefficients
Increase in the log odds of a one-unit increase in the regressor, holding values of all other regressors fixed.

2.14 Lab: Classification ([ISL] Ch 4.6)


```r
# The Stock Market Data

library(ISLR)
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

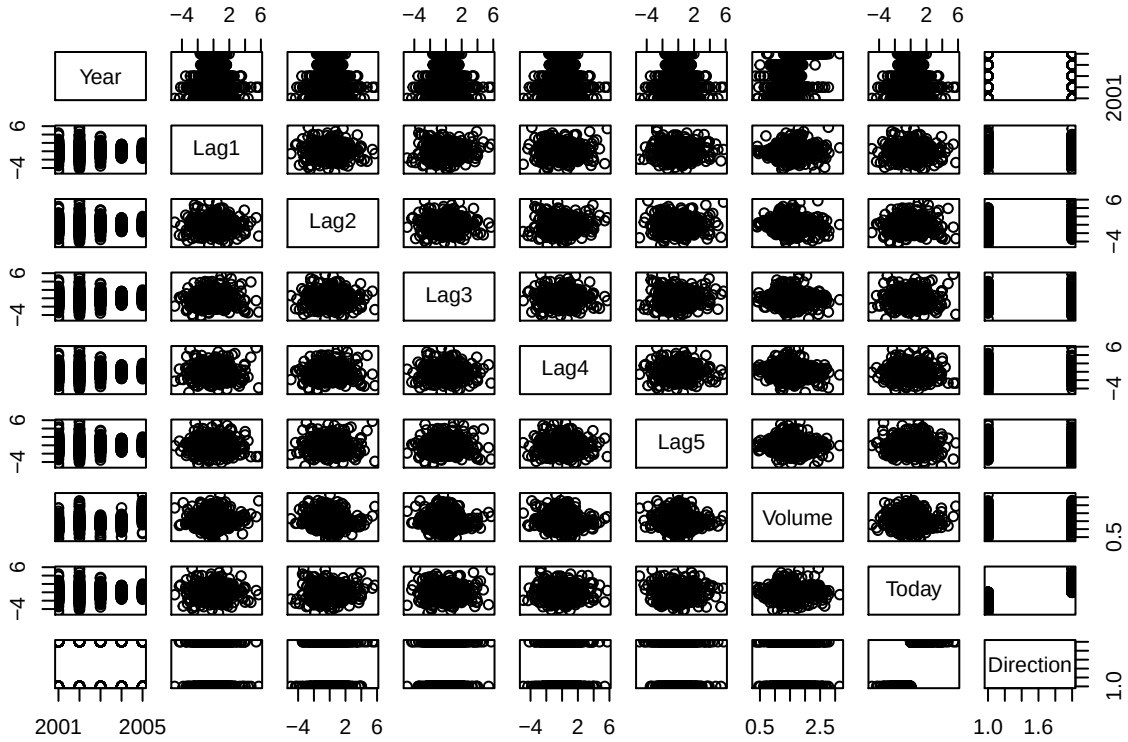
```r
dim(Smarket)
```

```
## [1] 1250    9
```

```r
summary(Smarket)
```

```
##       Year           Lag1                Lag2
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
##  Median :2003   Median : 0.039000   Median : 0.039000
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
##       Lag3                Lag4                Lag5
##  Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200
##  1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000
##  Median : 0.038500   Median : 0.038500   Median : 0.03850
##  Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561
##  3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700
##  Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300
##      Volume           Today           Direction
##  Min.   :0.3561   Min.   :-4.922000   Down:602
##  1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648
##  Median :1.4229   Median : 0.038500
##  Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   :3.1525   Max.   : 5.733000
```

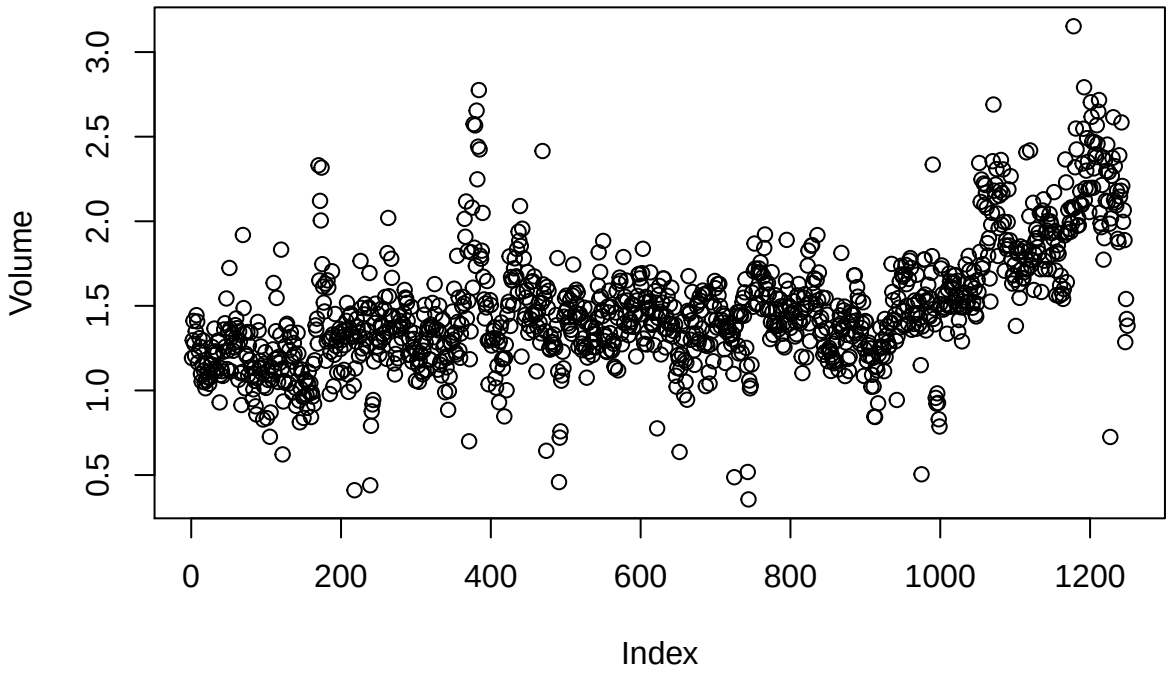
```r
pairs(Smarket)
```




```
# cor(Smarket)
cor(Smarket[,-9])
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

```
attach(Smarket)
plot(Volume)
```




```
# Logistic Regression

glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,family=binomial)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

```
coef(glm.fits)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938
##         Lag5       Volume
##  0.010313068  0.135440659
```

```
summary(glm.fits)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fits)$coef[,4]
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974
##      Volume
##   0.3924004
```

```
glm.probs=predict(glm.fits,type="response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509
##         8         9        10
## 0.5092292 0.5176135 0.4888378
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
glm.pred=rep("Down",1250)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```
(507+145)/1250
```

```
## [1] 0.5216
```

```
mean(glm.pred==Direction)
```

```
## [1] 0.5216
```

```
train=(Year<2005)
Smarket.2005=Smarket[!train,]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```
Direction.2005=Direction[!train]
glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,family=binomial,subset=train)
glm.probs=predict(glm.fits,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
##     Down   77 97
##     Up     34 44
```

```
mean(glm.pred==Direction.2005)
```

```
## [1] 0.4801587
```

```
mean(glm.pred!=Direction.2005)
```

```
## [1] 0.5198413
```

```
glm.fits=glm(Direction~Lag1+Lag2,data=Smarket,family=binomial,subset=train)
glm.probs=predict(glm.fits,Smarket.2005,type="response")
glm.pred=rep("Down",252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```
mean(glm.pred==Direction.2005)
```

```
## [1] 0.5595238
```

```
106/(106+76)
```

```
## [1] 0.5824176
```

```
predict(glm.fits,newdata=data.frame(Lag1=c(1.2,1.5),Lag2=c(1.1,-0.8)),type="response")
```

```
##         1         2
## 0.4791462 0.4960939
```

```
# Linear Discriminant Analysis

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

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
```

```
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

```
#plot(lda.fit)
lda.pred=predict(lda.fit, Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class=lda.pred$class
table(lda.class,Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
mean(lda.class==Direction.2005)
```

```
## [1] 0.5595238
```

```
sum(lda.pred$posterior[,1]>=.5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1]<.5)
```

```
## [1] 182
```

```
lda.pred$posterior[1:20,1]
```

```
##       999      1000      1001      1002      1003      1004      1005
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016
##      1006      1007      1008      1009      1010      1011      1012
## 0.4872861 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761
##      1013      1014      1015      1016      1017      1018
## 0.4744593 0.4799583 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up
## [15] Up   Up   Up   Down Up   Up
## Levels: Down Up
```

```
sum(lda.pred$posterior[,1]>.9)
```

```
## [1] 0
```

```
# Quadratic Discriminant Analysis

qda.fit=qda(Direction~Lag1+Lag2,data=Smarket,subset=train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
```

```
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```
qda.class=predict(qda.fit,Smarket.2005)$class
table(qda.class,Direction.2005)
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class==Direction.2005)
```

```
## [1] 0.5992063
```

```
# 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)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
(83+43)/252
```

```
## [1] 0.5
```

```
knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```
mean(knn.pred==Direction.2005)
```

```
## [1] 0.5357143
```

```
# An Application to Caravan Insurance Data

dim(Caravan)
```

```
## [1] 5822   86
```

```
attach(Caravan)
summary(Purchase)
```

```
##   No  Yes
## 5474  348
```

```
348/5822
```

```
## [1] 0.05977327
```

```
standardized.X=scale(Caravan[,-86])
var(Caravan[,1])
```

```
## [1] 165.0378
```

```
var(Caravan[,2])
```

```
## [1] 0.1647078
```

```
var(standardized.X[,1])
```

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

```
var(standardized.X[,2])
```

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

```
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)
```

```
## [1] 0.118
```

```
mean(test.Y!="No")
```

```
## [1] 0.059
```

```
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```
9/(68+9)
```

```
## [1] 0.1168831
```

```
knn.pred=knn(train.X,test.X,train.Y,k=3)
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  920  54
##      Yes  21   5
```

```
5/26
```

```
## [1] 0.1923077
```

```
knn.pred=knn(train.X,test.X,train.Y,k=5)
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  930  55
##      Yes  11   4
```

```
4/15
```

```
## [1] 0.2666667
```

```
glm.fits=glm(Purchase~.,data=Caravan,family=binomial,subset=-test)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm.probs=predict(glm.fits,Caravan[test,],type="response")
glm.pred=rep("No",1000)
glm.pred[glm.probs>.5]="Yes"
table(glm.pred,test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  934  59
##      Yes   7   0
```

```
glm.pred=rep("No",1000)
glm.pred[glm.probs>.25]="Yes"
table(glm.pred,test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  919  48
##      Yes  22  11
```

```
11/(22+11)
```

```
## [1] 0.3333333
```

2.15 Cross-validation ([ISL] Ch. 5.1)

Explain cross-validation

A class of methods that estimate the test error rate by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations

Explain the validation set approach

Involves randomly dividing the available set of observations into two parts, a training set and a validation set or hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate provides an estimate of the test error rate.

List two drawbacks of the validation set approach

1. The validation estimate of the test error rate can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.

2. Only a subset of the observations are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set.

Explain leave-one-out cross-validation

First, a single observation is used for the validation set, and the remaining observations make up the training set. The statistical learning method is fit on the $n-1$ training observations, a prediction is made for the excluded observation and its squared error is computed – note that even though unbiased for the test error, it is a poor estimate because it is highly variable since it is based upon a single observation. We can repeat the procedure by selecting another point for the validation data, training the statistical learning procedure on the other $n-1$ observations and computing another squared error term. Repeating this approach n times produces n squared errors. The LOOCV estimate for the test MSE is the average of these n test error estimates:

List two advantages of LOOCV over the validation set approach

1. It has far less bias. In LOOCV, we repeatedly fit the statistical learning method using training sets that contain $n-1$ observations, almost as many as are in the entire data set, hence tends not to overestimate the test error rate.

2. Performing LOOCV multiple times will always yield the same results: there is no randomness in the training/validation set splits.

Describe a potential drawback, and shortcut for least squares regression

LOOCV has the potential to be very time consuming to implement, since the model has to be fit n times. With least squares linear or polynomial regression, the following formula makes the cost of LOOCV the same as that of a single model fit: $CV(n) = \frac{1}{n}\sum_{i=1}^{n}(\frac{y_i=\hat{y}_i}{1-h_i})^2$

where $\hat{y}_i$ is the ith fitted value from the original least squares fit, and h_i is the leverage of the ith data point which reflects the amount that an observation influences its own fit. This is like the ordinary MSE, except the ith residual is divided by $1-h_i$, sothat the residuals for high-leverage points are inflated by exactly the right amount for this equality to hold.

Explain k-fold cross-validation

This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the observations in the remaining $k-1$ folds. The mean squared error is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, and the k-fold CV estimate is computed by averaging these values.

Explain the bias-variance trade-off associated with thechoice of k in k-fold cross-validation

LOOCV will give approximately unbiased estimates of the test error, since each training set contains almost as many as the number of observations in the full data set. Hence from the perspective of bias reduction LOOCV is preferred over k-fold CV, where each training set contains fewer observations. However, the outputs of the n fitted LOOCV are highly (positively) correlated with each other. In contrast, we are averaging the outputs of k fitted k-fold models that are somewhat less correlated with each other (when $k < n$), since the overlap between the training sets in each model is smaller. Since the mean of many highly correlated quantities has higher variance than does the mean of many quantities that are not as highly correlated, the test error estimate resulting from LOOCV tends to have higher variance. Using k = 5 or 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

Explain the use of CV curves for model selection

We can perform cross-validation on a number of statistical learning methods, or on a single method using different levels of flexibility, in order to identify the method that results in the lowest test error. For this purpose, the location of the minimum point in the estimated test MSE curve is important, but the actual value of the estimated test MSE is not. Despite the fact that they sometimes underestimate the true test MSE, the CV curves come close to identifying the correct level of flexibility – that is, the flexibility level corresponding to the smallest test MSE.

2.16 Lab: Cross-Validation ([ISL] Ch 5.3)


```r
# 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)
```

```
## [1] 23.26601
```

```r
lm.fit2=lm(mpg~poly(horsepower,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
```

```
## [1] 18.71646
```

```r
lm.fit3=lm(mpg~poly(horsepower,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)
```

```
## [1] 18.79401
```

```r
set.seed(2)
train=sample(392,196)
lm.fit=lm(mpg~horsepower,subset=train)
mean((mpg-predict(lm.fit,Auto))[-train]^2)
```

```
## [1] 25.72651
```

```r
lm.fit2=lm(mpg~poly(horsepower,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
```

```
## [1] 20.43036
```

```r
lm.fit3=lm(mpg~poly(horsepower,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)
```

```
## [1] 20.38533
```

```r
# Leave-One-Out Cross-Validation

glm.fit=glm(mpg~horsepower,data=Auto)
coef(glm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```r
lm.fit=lm(mpg~horsepower,data=Auto)
coef(lm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```r
library(boot)
glm.fit=glm(mpg~horsepower,data=Auto)
cv.err=cv.glm(Auto,glm.fit)
cv.err$delta
```

```
## [1] 24.23151 24.23114
```

```
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
```

```
## [1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

```
# 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
```

```
##  [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061
##  [8] 19.14666 18.87013 20.95520
```

```
# 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)
```

```
## [1] 0.5758321
```

```
set.seed(1)
alpha.fn(Portfolio,sample(100,100,replace=T))
```

```
## [1] 0.7368375
```

```
boot(Portfolio,alpha.fn,R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.5758321 -0.001695873  0.09366347
```

```
# 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)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```
set.seed(1)
boot.fn(Auto,sample(392,392,replace=T))
```

```
## (Intercept)  horsepower
##  40.3404517  -0.1634868
```

```
boot.fn(Auto,sample(392,392,replace=T))
```

```
## (Intercept)  horsepower
##  40.1186906  -0.1577063
```

```
boot(Auto,boot.fn,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0544513229 0.841289790
## t2* -0.1578447 -0.0006170901 0.007343073
```

```
summary(lm(mpg~horsepower,data=Auto))$coef
```

```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) 39.9358610 0.717498656  55.65984 1.220362e-187
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-81
```

```
boot.fn=function(data,index)
 coefficients(lm(mpg~horsepower+I(horsepower^2),data=data,subset=index))
set.seed(1)
boot(Auto,boot.fn,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 56.900099702  3.511640e-02 2.0300222526
## t2* -0.466189630 -7.080834e-04 0.0324241984
## t3*  0.001230536  2.840324e-06 0.0001172164
```

```
summary(lm(mpg~horsepower+I(horsepower^2),data=Auto))$coef
```

```
##                     Estimate   Std. Error   t value      Pr(>|t|)
## (Intercept)     56.900099702 1.8004268063  31.60367 1.740911e-109
```

```
## horsepower      -0.466189630 0.0311246171 -14.97816  2.289429e-40
## I(horsepower^2)  0.001230536 0.0001220759  10.08009  2.196340e-21
```

2.17 Subset selection ([ISL] Ch. 6.1)

Define best subset selection.

To perform best subset selection, we fit a separate least squares regression for each possible combination of the p predictors. That is, we fit all p models that contain exactly one predictor, all $p^2 = p(p-1)/2$ models that contain exactly two predictors, and so forth. We then look at all of the resulting 2^p models, with the goal of identifying the one that is best.

List the steps used in best subset selection.

1. Let M_0 denote the null model , which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, ...p$:
   - Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   - Pick the best among these $\binom{p}{k}$ models, and call it M_k. Here best is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $M_0, ..., M_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

Define deviance.

Deviance is a measure that plays the role of RSS for a broader class of models. The deviance is negative two times the maximized log-likelihood; the smaller the deviance, the better the fit.

Describe forward stepwise selection and backward stepwise selection.

Forward and backward stepwise selection are computationally efficient alternatives to best subset selection. While the best subset selection procedure considers all 2^p possible models containing subsets of the p predictors, forward stepwise considers a much smaller set of models. It begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model. Backward stepwise selection begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time.

List the steps used in forward stepwise selection and backward stepwise selection.

Forward stepwise selection:

1. Let M_0 denote the null model, which contains no predictors.
2. For $k = 1, 2, ...p - 1$:
   - Choose the best among these $p-k$ models that augment the predictors in M_k with one additional predictor.
   - Choose the best among these $p - k$ models, and call it M_{k+1}. Here best is defined as having the smallest RSS or highest R^2.
3. Select a single best model from among $M_0, ..., M_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

Backward stepwise selection:

1. Let M_p denote the full model, which contains all p predictors.

2. For $k = p, p-1, ..., 1$:
   - Choose all k models that contain all but one of the predictors for a total of $k-1$ predictors.
   - Choose the best among these k models, and call it M_{k-1}. Here best is defined as having the smallest RSS or highest R^2.
3. Select a single best model from among $M_0, ..., M_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2 .

Recognize and apply the equations for C_p , Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and adjusted R^2.

Techniques for adjusting the training error for the model size to select among a set of models with different numbers of variables, d:

- $C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$, where σ^2 is an estimate of the variance of the error associated with each response measurement, typically estimated using the full model containing all predictors. Adds a penalty of $2d\hat{\sigma}^2$ to the training RSS in order to adjust for the fact that the training error tends to underestimate the test error. The penalty increases as the number of predictors in the model increases; this is intended to adjust for the corresponding decrease in training RSS.
- Akaike Information Criterion: The AIC criterion is defined for a large class of models fit by maximum likelihood. In the case of the standard linear regression model with Gaussian errors, maximum likelihood and least squares are the same thing, hence C_p and AIC are proportional to each other. $AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$
- Bayesian Information Criterion: BIC is derived from a Bayesian point of view, but ends up looking similar to C p (and AIC) as well. BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $log(n)d\hat{\sigma}^2$ term, where n is the number of observations. Since $\log n > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models. $BIC = \frac{1}{n\hat{\sigma}^2}(RSS + \log(n)d\hat{\sigma}^2)$
- adjusted $R^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$. The usual R^2 always increases (since RSS always decreases) as more variables are added. The intuition behind the adjusted R^2 is that once all of the correct variables have been included in the model, adding noise variables will lead to an increase in $\frac{RSS}{n-d-1}$ (due to the presence of d in the denominator) and consequently a decrease in the statistic. Despite its popularity, and even though it is quite intuitive, the adjusted R^2 is not as well motivated in statistical theory as AIC, BIC, and C_p.

2.18 Shrinkage methods ([ISL] Ch. 6.2)

Define ridge regression, tuning parameter, and shrinkage penalty.

- Ridge regression: Very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity: $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$.
- Shrinkage penalty: The second term, $\lambda\sum_{j=1}^{p}\beta_j^2$, is small when the coefficients are close to zero, and so it has the effect of shrinking the estimates towards zero.
- Tuning parameter: λ serves to control the relative impact of these RSS and shrinkage penalty terms on the coefficient estimates in a ridge regression model. When $\lambda = 0$, the penalty term has no effect,

and ridge regression will produce the least squares estimates. However, as $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero. Selecting a good value for λ is critical, such as by using cross-validation.

Define l2 norm and scale equivalent.

- l_2 norm: $||\beta||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$, which measures the distance of β from zero.
- Scale equivalent: When multiplying a predictor variable X_j by a constant c simply leads to a scaling of the coefficient estimates $\hat{\beta}_j$ by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.

Define standardizing the predictors.

Using the formula $\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\overline{x}_j)^2}}$ so that the predictor variables are all on the same scale. The denominator is the estimated standard deviation of the jth predictor. Consequently, all of the standardized predictors will have a standard deviation of one. As a result the final fit of the ridge regression model will not depend on the scale on which the predictors are measured.

Describe bias-variance tradeoff.

The test mean squared error (MSE) is a function of the variance plus the squared bias. The shrinkage of the ridge coefficient estimates leads to a substantial reduction in the variance of the predictions, at the expense of a slight increase in bias.

Describe the ridge regression.

As the tuning parameter λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. At the least squares coefficient estimates, which correspond to ridge regression with $\lambda = 0$, the variance is high but there is no bias. As the tuning parameter increases, the shrinkage of the ridge coefficient estimates leads to a substantial reduction in the variance of the predictions, at the expense of a slight increase in bias. For an intermediate value of λ, the MSE is considerably lower.

Describe how ridge regression improves upon least squares.

Ridge regression's advantage over least squares is rooted in the bias-variance trade-off. In situations where the relationship between the response and the predictors is close to linear, the least squares estimates will have low bias but may have high variance. This means that a small change in the training data can cause a large change in the least squares coefficient estimates. In particular, when the number of variables p is almost as large as the number of observations n, the least squares estimates will be extremely variable. And if $p > n$, then the least squares estimates do not even have a unique solution, whereas ridge regression can still perform well by trading off a small increase in bias for a large decrease in variance. Hence, ridge regression works best in situations where the least squares estimates have high variance.

Describe the advantage of Lasso over the ridge regression.

Ridge regression will shrink all of the coefficients towards zero, but will include all p predictors in the final model. Lasso forces some of the coefficient estimates to be exactly equal to zero when the tuning parameter is sufficiently large. Depending on the tuning parameter, lasso can produce a model involving any number of variables, whereas ridge regression will always include all of the variables in the model, although the magnitude of the coefficient estimates will depend on λ. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression.

Define a sparse model.

Sparse models involve only a subset of the variables.

Describe the variable selection property of the Lasso.

When the tuning parameter is sufficiently large, some of the coefficient estimates are forced to be exactly equal to zero. This can also be visualized by plotting the constraint functions for the lasso which are polytopes (or squares when $p = 2$ or polyhedrons when $p = 3$). Coefficient estimates are given by the first point at which an ellipse of the error function contacts the constraint region. The lasso constraint has corners at each of the axes, and so the ellipse will often intersect the constraint region at an axis. When this occurs, one of the coefficients will equal zero. In higher dimensions, many of the coefficient estimates may equal zero simultaneously. Note that since ridge regression has a circular constraint with no sharp points, this intersection will not generally occur on an axis, and so the ridge regression coefficient estimates will be exclusively non-zero. Hence, much like best subset selection, the lasso performs variable selection.

Compare the Lasso to the Ridge regression.

The lasso coefficients $\hat{\beta}_\lambda^L$ minimize $\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$. Compared to ridge regression, the only difference is that the β_j^2 term has been replaced by $|\beta_j|$ in the lasso penalty. Lasso uses an l_1 penalty instead of an l_2 penalty. The l_1 norm of a coefficient vector is given by $||\beta||_1 = \sum |\beta_j|$. Unlike ridge regression, the lasso performs variable selection, and hence results in models that are easier to interpret. In terms of MSE, neither ridge regression nor the lasso will universally dominate the other. In general, one might expect the lasso to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or that equal zero. Ridge regression will perform better when the response is a function of many predictors, all with coefficients of roughly equal size. Ridge regression and the lasso can also be viewed through a Bayesian lens. If the prior distribution of the coefficients is Gaussian distribution, then the posterior mode is given by the ridge regression solution. If the prior is is a double-exponential (Laplace), then the posterior mode for is the lasso solution.

Describe how to select the tuning parameter.

We choose a grid of λ values, and compute the cross-validation error for each value. We then select the tuning parameter value for which the cross-validation error is smallest. Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.

2.19 Dimension reduction methods ([ISL] Ch. 6.3)

Define dimension reduction and linear combination.

Dimension reduction methods are a class of techniques that transform the p predictors and then fit a least squares model using the M transformed variables, where the problem is reduced to estimating fewer $p+1 < M+1$ coefficients. Linear combinations of our original p predictors are represented by $Z_1, Z_2, ..., Z_M$, where $M < p$: $Z_m = \sum_{j=1}^p \phi_{jm} X_j$ for some constants $\phi_{1m}, \phi_{2m}, ..., \phi_{pm}$, $m = 1, ..., M$.

Describe principal component analysis.

Principal components analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables. The first principal component has been chosen so that the projected observations are as close as possible to the original observations, which is also the direction along which the data vary the most. Out of every possible linear combination of the variables, this particular linear combination yields the highest variance. The second principal component is a linear combination of the variables that is uncorrelated with the first, and has largest variance subject to this constraint. One can then construct up to p distinct principal components.

Describe principal component regression.
This approach involves constructing the first M principal components, $Z_1, Z_2, ..., Z_M$, and then using these components as the predictors in a linear regression model that is fit using least squares. The key idea is that often a small number of principal components suffice to explain most of the variability in the data, and assumes that the directions in which $X_1, ..., X_p$ show the most variation are the directions that are associated with Y (though this assumption is not guaranteed)

Describe partial least squares.
PLS, a supervised alternative to PCR makes use of the response Y in order to identify new features that not only approximate the old features well, but also that are related to the response. After standardizing the p predictors, PLS computes the first direction Z_1 by setting each ϕ_{j1} in equal to the coefficient from the simple linear regression of Y onto X_j, which isis proportional to the correlation between Y and X_j. Hence, PLS places the highest weight on the variables that are most strongly related to the response. The residuals, can be interpreted as the remaining information that has not been explained by the first PLS direction, are then used in exactly the same fashion to compute the next direction Z_2. This iterative approach can be repeated M times to identify multiple PLS components. Finally, at the end of this procedure, we use least squares to fit a linear model to predict Y using $Z_1, ..., Z_M$ in exactly the same fashion as for PCR.

2.20 Considerations in high dimensions ([ISL] Ch. 6.4)

Define low-dimensional and high-dimensional data.
In a low-dimensional setting, the number of observations is much greater than the number of features. In a high-dimensional setting, the number of features is larger than the number of observations.

Describe what goes wrong in high dimensions.
Regardless of whether or not there truly is a relationship between the features and the response, least squares will yield a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero. This is problematic because this perfect fit, even though the features are completely unrelated to the response, will almost certainly lead to overfitting of the data, as the resulting model will perform extremely poorly on an independent test set,

Describe regression in high dimensions.
Methods such as forward stepwise selection, ridge regression, the lasso, and principal components regression, are particularly useful for performing regression in the high-dimensional setting. Essentially, these approaches avoid overfitting by using a less flexible fitting approach than least squares. Regularization or shrinkage plays a key role in high-dimensional problems, and appropriate tuning parameter selection is crucial for good predictive performance,

Define curse of dimensionality.
The test error tends to increase as the dimensionality of the problem (i.e. the number of features or predictors) increases, unless the additional features are truly associated with the response. Noise features exacerbating the risk of overfitting (since noise features may be assigned nonzero coefficients due to chance associations with the response on the training set) without any potential upside in terms of improved test set error. Even if they are relevant, the variance incurred in fitting their coefficients may outweigh the reduction in bias that they bring.

2.21 Lab: Model Selection ([ISL] Ch 6.5-7)


```r
# Best Subset Selection
library(ISLR)
#fix(Hitters)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
```

```r
dim(Hitters)
```

```
## [1] 322  20
```

```r
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```r
Hitters=na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```r
sum(is.na(Hitters))
```

```
## [1] 0
```

```r
library(leaps)
regfit.full=regsubsets(Salary~.,Hitters)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., Hitters)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "
```

```
## 2  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "
## 3  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "
## 4  ( 1 ) " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "
## 5  ( 1 ) "*"   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    " "
## 6  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    " "
## 7  ( 1 ) " "   "*"  " "   " "  " " " " "*"   " "   "*"    "*"   "*"    " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   "*"    "*"
##          CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) "*"  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "  " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) " "  "*"    " "     "*"       "*"     " "     " "    " "
```

```
regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=19)
reg.summary=summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
reg.summary$rsq
```

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

```
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)
```

```
## [1] 11
```

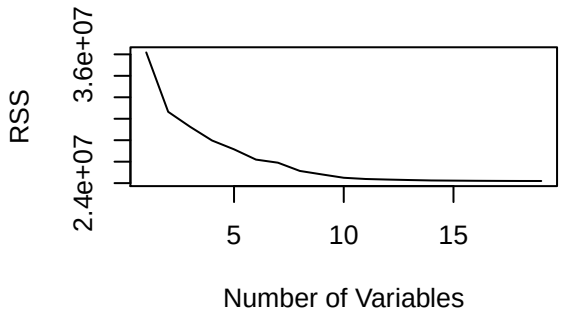
```
points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
which.min(reg.summary$cp)
```

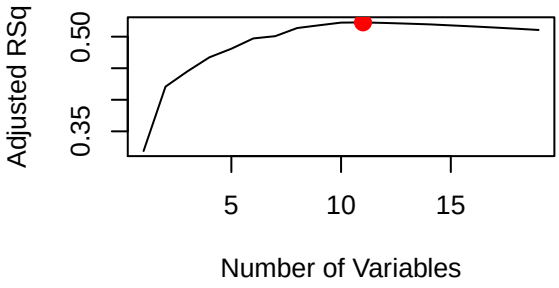
```
## [1] 10
```

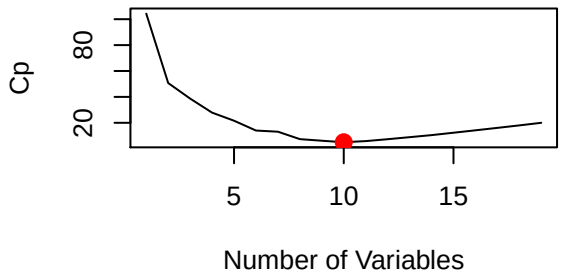
```
points(10,reg.summary$cp[10],col="red",cex=2,pch=20)
which.min(reg.summary$bic)
```

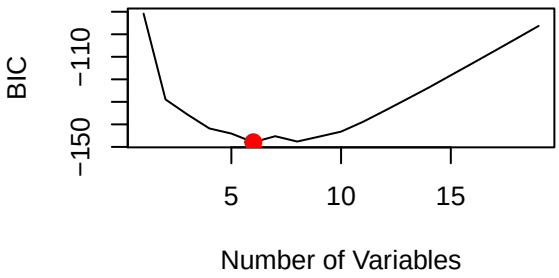
```
## [1] 6
```

```
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(6,reg.summary$bic[6],col="red",cex=2,pch=20)
```

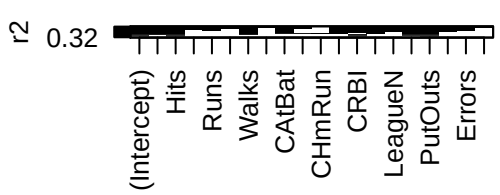


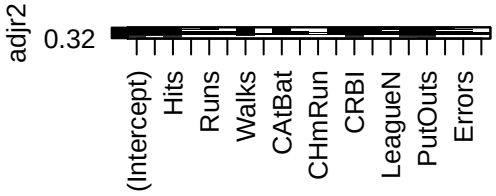


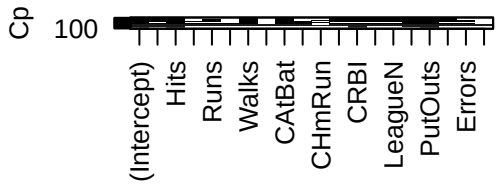


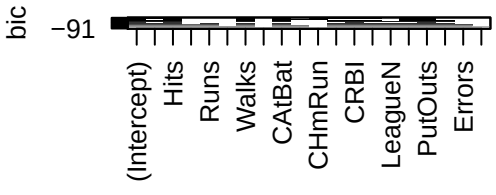



```
plot(regfit.full,scale="r2")
plot(regfit.full,scale="adjr2")
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
```










```
coef(regfit.full,6)
```

```
##  (Intercept)        AtBat         Hits        Walks         CRBI
##   91.5117981   -1.8685892    7.6043976    3.6976468    0.6430169
##    DivisionW      PutOuts
```

```
## -122.9515338    0.2643076
```

```
# Forward and Backward Stepwise Selection

regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: forward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 1  ( 1 )  " "   " "  " "   " "  " " " " " "   " "   " "   " "    " "    " "
## 2  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "    " "
## 3  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "    " "
## 4  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "   " "    " "    " "
## 5  ( 1 )  "*"   "*"  " "   " "  " " " " " "   " "   " "   " "    " "    " "
## 6  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    " "    " "
## 7  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    " "    " "
## 8  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "   " "    " "    "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   "*"   " "    " "    "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   "*"   " "    " "    "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " " " "*"   " "   "*"   " "    " "    "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " " " "*"   " "   "*"   " "    " "    "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " " " "*"   " "   "*"   " "    " "    "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " " " "*"   " "   "*"   " "    " "    "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " " " "*"   " "   "*"   "*"    " "    "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" " " "*"   " "   "*"   "*"    " "    "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   " "   "*"   "*"    " "    "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"   "*"    " "    "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*" "*"   "*"   "*"   "*"    "*"    "*"
##           CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  "*"  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  "*"  " "    " "     " "       " "     " "     " "    " "
```

```
## 3  ( 1 )  "*"  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  "*"  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 )  "*"  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  "*"  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  "*"  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"  "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: backward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 1  ( 1 )  " "   " "  " "   " "  " " " " " "   " "   " "    " "   " "    "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    "*"
## 4  ( 1 )  "*"   "*"  " "   " "  " " " " " "   " "   " "    " "   " "    "*"
## 5  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    "*"
## 7  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    "*"
## 8  ( 1 )  "*"   "*"  " "   " "  " " " " "*"   " "   " "    " "   " "    "*"
```

```
## 9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"
##           CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "  " "    " "     " "       "*"     " "     " "    " "
## 5  ( 1 )  " "  " "    " "     " "       "*"     " "     " "    " "
## 6  ( 1 )  " "  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  " "  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"  "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
coef(regfit.full,7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073
##       CHmRun    DivisionW      PutOuts
##    1.4420538 -129.9866432    0.2366813
```

```
coef(regfit.fwd,7)
```

```
##  (Intercept)        AtBat         Hits        Walks         CRBI
##  109.7873062   -1.9588851    7.4498772    4.9131401    0.8537622
##       CWalks    DivisionW      PutOuts
##   -0.3053070 -127.1223928    0.2533404
```

```
coef(regfit.bwd,7)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns
##  105.6487488   -1.9762838    6.7574914    6.0558691    1.1293095
##       CWalks    DivisionW      PutOuts
##   -0.7163346 -116.1692169    0.3028847
```

```
# 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
```

```
##  [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0
##  [8] 136191.4 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2
## [15] 141508.2 142164.4 141767.4 142339.6 142238.2
```

```
which.min(val.errors)
```

```
## [1] 7
```

```
coef(regfit.best,10)
```

```
##  (Intercept)        AtBat         Hits        HmRun        Walks
##   71.8074075   -1.5038124    5.9130470  -11.5241809    8.4349759
##       CAtBat        CRuns         CRBI       CWalks    DivisionW
##   -0.1654850    1.7064330    0.7903694   -0.9107515 -109.5616997
##      PutOuts
##    0.2426078
```

```
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)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat
##  162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798
##        CRuns         CRBI       CWalks    DivisionW      PutOuts
##    1.4082490    0.7743122   -0.8308264 -112.3800575    0.2973726
##      Assists
##    0.2831680
```

```
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
```

```
##        1        2        3        4        5        6        7        8
## 149821.1 130922.0 139127.0 131028.8 131050.2 119538.6 124286.1 113580.0
##        9       10       11       12       13       14       15       16
## 115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3
##       17       18       19
## 120084.8 120085.8 120403.5
```

```
par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
reg.best=regsubsets(Salary~.,data=Hitters, nvmax=19)
coef(reg.best,11)
```

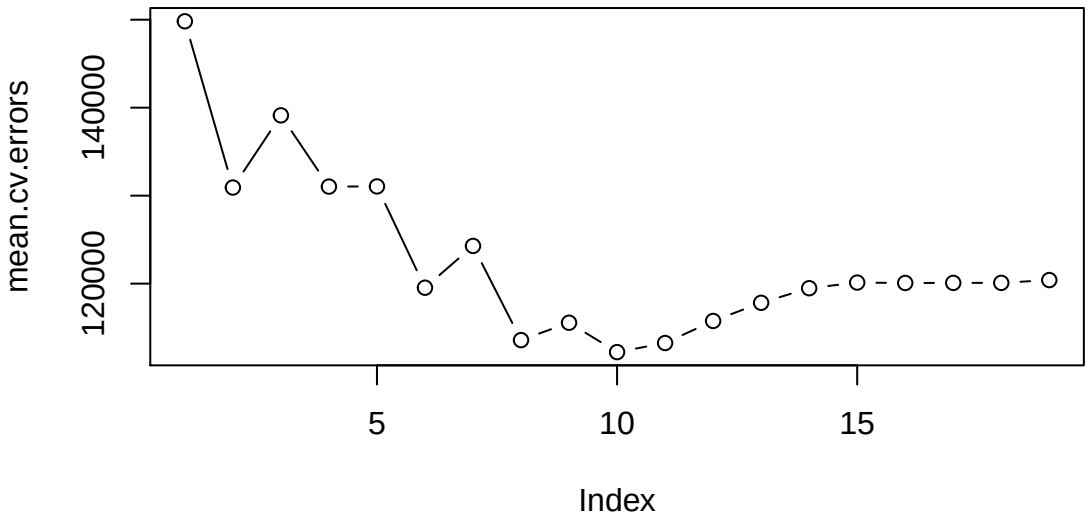
```
##  (Intercept)        AtBat         Hits        Walks       CAtBat
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914
##        CRuns         CRBI       CWalks      LeagueN    DivisionW
##    1.4553310    0.7852528   -0.8228559   43.1116152 -111.1460252
##      PutOuts      Assists
##    0.2894087    0.2688277
```

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

# Ridge Regression

library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```




```
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
```

```
## [1]  20 100
```

```
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 407.356050200   0.036957182   0.138180344   0.524629976   0.230701523
##           RBI         Walks         Years        CAtBat         CHits
##   0.239841459   0.289618741   1.107702929   0.003131815   0.011653637
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.087545670   0.023379882   0.024138320   0.025015421   0.085028114
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
##  -6.215440973   0.016482577   0.002612988  -0.020502690   0.301433531
```

```
sqrt(sum(coef(ridge.mod)[-1,50]^2))
```

```
## [1] 6.360612
```

```
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##  54.32519950   0.11211115   0.65622409   1.17980910   0.93769713
##          RBI        Walks        Years       CAtBat        CHits
##   0.84718546   1.31987948   2.59640425   0.01083413   0.04674557
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.33777318   0.09355528   0.09780402   0.07189612  13.68370191
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -54.65877750   0.11852289   0.01606037  -0.70358655   8.61181213
```

```
sqrt(sum(coef(ridge.mod)[-1,60]^2))
```

```
## [1] 57.11001
```

```
predict(ridge.mod,s=50,type="coefficients")[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  4.876610e+01 -3.580999e-01  1.969359e+00 -1.278248e+00  1.145892e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.038292e-01  2.716186e+00 -6.218319e+00  5.447837e-03  1.064895e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  6.244860e-01  2.214985e-01  2.186914e-01 -1.500245e-01  4.592589e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.182011e+02  2.502322e-01  1.215665e-01 -3.278600e+00 -9.496680e+00
```

```
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)
```

```
## [1] 142199.2
```

```
mean((mean(y[train])-y.test)^2)
```

```
## [1] 224669.9
```

```
ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 224669.8
```

```
ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train])
mean((ridge.pred-y.test)^2)
```

```
## [1] 168588.6
```

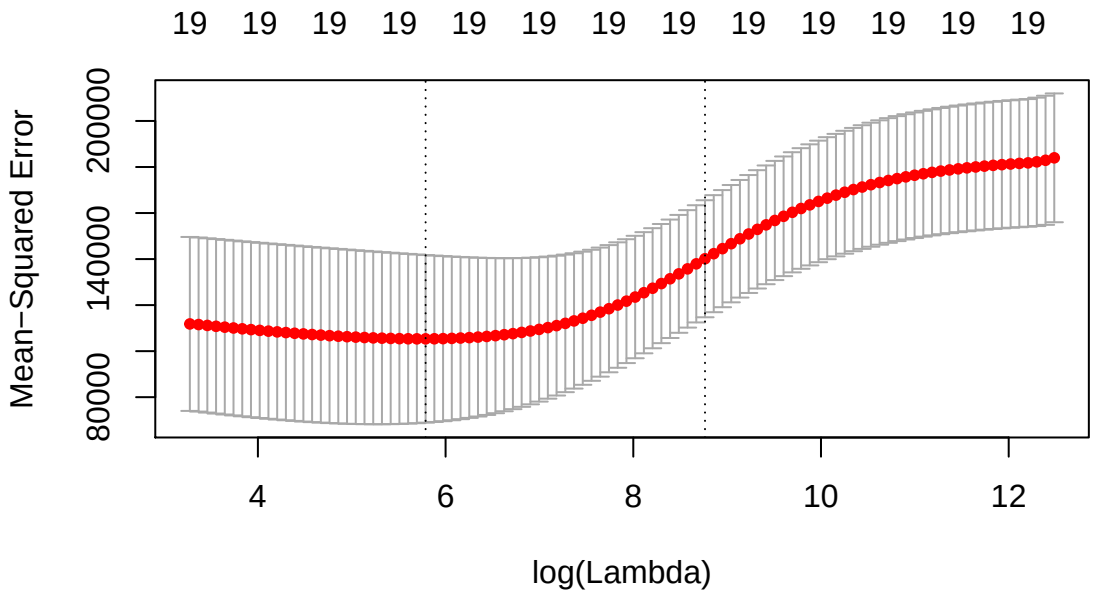
```
lm(y~x, subset=train)
```

```
##
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
## (Intercept)       xAtBat        xHits       xHmRun        xRuns
##    274.0145      -0.3521      -1.6377       5.8145       1.5424
##        xRBI       xWalks       xYears      xCAtBat       xCHits
##      1.1243       3.7287     -16.3773      -0.6412       3.1632
##     xCHmRun       xCRuns        xCRBI      xCWalks     xLeagueN
##      3.4008      -0.9739      -0.6005       0.3379     119.1486
##  xDivisionW     xPutOuts     xAssists      xErrors  xNewLeagueN
##   -144.0831       0.1976       0.6804      -4.7128     -71.0951
```

```
predict(ridge.mod,s=0,exact=T,type="coefficients",x=x[train,],y=y[train])[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##  274.0200994   -0.3521900   -1.6371383    5.8146692    1.5423361
##          RBI        Walks        Years       CAtBat        CHits
##    1.1241837    3.7288406  -16.3795195   -0.6411235    3.1629444
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##    3.4005281   -0.9739405   -0.6003976    0.3378422  119.1434637
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -144.0853061    0.1976300    0.6804200   -4.7127879  -71.0898914
```

```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```




```
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 326.0828
```

```
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

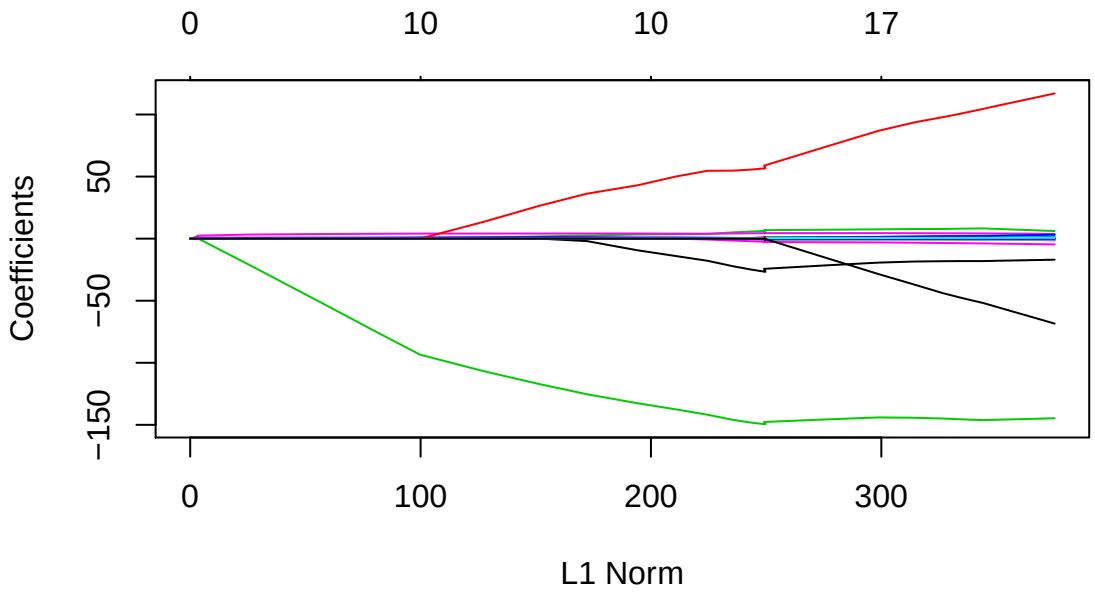
```
## [1] 139856.6
```

```
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlam)[1:20,]
```

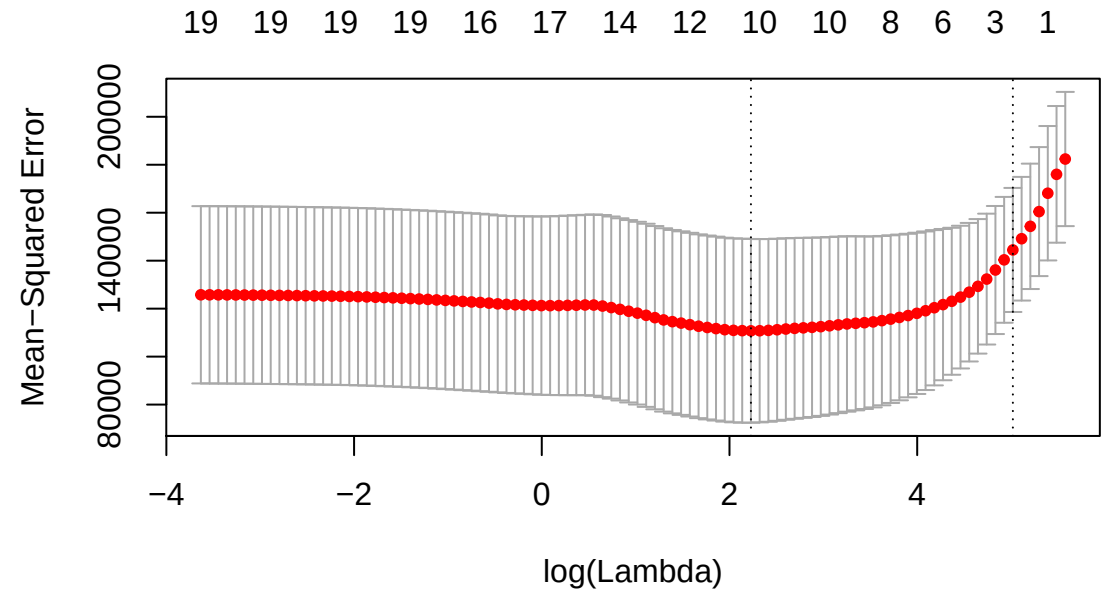
```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##  15.44383135   0.07715547   0.85911581   0.60103107   1.06369007
##          RBI        Walks        Years       CAtBat        CHits
##   0.87936105   1.62444616   1.35254780   0.01134999   0.05746654
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.40680157   0.11456224   0.12116504   0.05299202  22.09143189
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -79.04032637   0.16619903   0.02941950  -1.36092945   9.12487767
```

```
# The Lasso

lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```




```
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)
```

```
## [1] 143673.6
```

```
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##    1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
```

```
##    0.00000000    2.29192406   -0.33806109    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI       CWalks       LeagueN
##    0.02825013    0.21628385    0.41712537    0.00000000   20.28615023
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -116.16755870    0.23752385    0.00000000   -0.85629148    0.00000000
```

```
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)         AtBat          Hits         Walks         Years
##    1.27479059   -0.05497143    2.18034583    2.29192406   -0.33806109
##        CHmRun         CRuns          CRBI       LeagueN     DivisionW
##    0.02825013    0.21628385    0.41712537   20.28615023 -116.16755870
##       PutOuts        Errors
##    0.23752385   -0.85629148
```

```
# Principal Components Regression

library(pls)
```

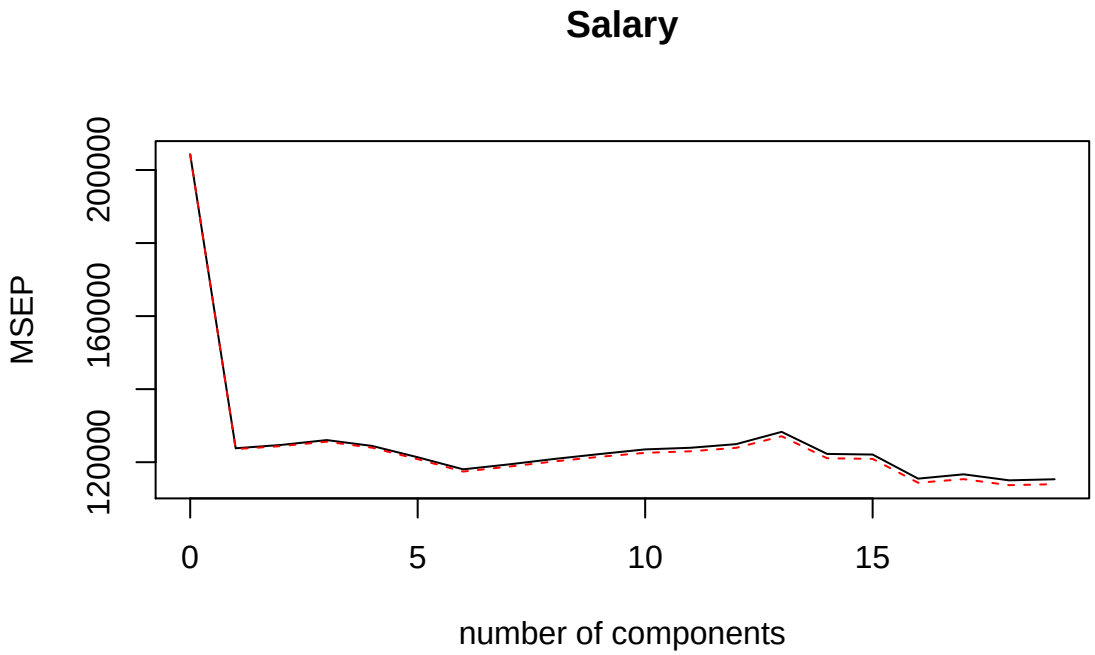
```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(2)
pcr.fit=pcr(Salary~., data=Hitters,scale=TRUE,validation="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X         94.96    96.28     97.26     97.98     98.65     99.15     99.47
## Salary    46.75    46.86     47.76     47.82     47.85     48.10     50.40
##         15 comps  16 comps  17 comps  18 comps  19 comps
## X          99.75     99.89     99.97     99.99    100.00
## Salary     50.55     53.01     53.85     54.61     54.61
```

```
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)
```

```
## [1] 140751.3
```

```
pcr.fit=pcr(y~x,scale=TRUE,ncomp=7)
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 7
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X    38.31    60.16    70.84    79.03    84.29    88.63    92.26
## y    40.63    41.58    42.17    43.22    44.90    46.48    46.69
```

```
# Partial Least Squares

set.seed(1)
pls.fit=plsr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
## adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
## adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        348.4     349.1     350.0     344.2     344.5     345.0
## adjCV     344.2     345.0     345.9     340.4     340.6     341.1
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X         90.71    93.17     96.05     97.08     97.61     97.97     98.70
## Salary    55.66    55.95     56.12     56.47     56.68     57.37     57.76
##         15 comps  16 comps  17 comps  18 comps  19 comps
## X          99.12     99.61     99.70     99.95    100.00
## Salary     58.08     58.17     58.49     58.56     58.62
```

```
validationplot(pls.fit,val.type="MSEP")
```


```
pls.pred=predict(pls.fit,x[test,],ncomp=2)
mean((pls.pred-y.test)^2)
```

```
## [1] 145367.7
```

```
pls.fit=plsr(Salary~., data=Hitters,scale=TRUE,ncomp=2)
summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##         1 comps  2 comps
## X         38.08    51.03
## Salary    43.05    46.40
```

3. Topic: Time Series Models

The Candidate will understand key concepts concerning regression-based time series models.

1. Define and explain the concepts and components of stochastic time series processes, including random walks, stationarity, and autocorrelation.
2. Describe specific time series models, including, exponential smoothing, autoregressive, and autoregressive conditionally heteroskedastic models.
3. Calculate and interpret predicted values and confidence intervals.

3.1 Modeling Trends ([REG] Ch. 7:1-6)

Define time series, longtitudinal, cross-sectional and panel data

Longitudinal data are measurements of a process that evolves over time. A single measurement of a process yields a variable over time, and referred to as a time series. Data that are not ordered are called cross-sectional. Longitudinal data examine a cross-section of entities and their evolution over time. This type of data is also known as panel data.

Define causal models and spurous relationships.

The apparent relationship between two series that were generated independently, because both are related to a trend over time, is said to be spurious. regression models of the form $y_t = \beta_0 + \beta_1 x_t + \epsilon_t$ are known as causal models, where it is assumed that economic theory provides the information needed to specify the causal relationship (x "causes" y)

List 3 components of time series for forecasting.

Time series can be conveniently decomposed a series into three types of patterns: trends in time; seasonal; and random, or irregular, patterns. A future values of the series can then be forecast by extrapolating each of the three patterns. The trend is that part of a series that corresponds to a long-term, slow evolution of the series. This is the most important part for long-term forecasts. The seasonal part of the series corresponds to aspects that repeat itself periodically, say, over a year. The irregular patterns of a series are short-term movements that are typically harder to anticipate.

Define seasonal adjustment.

Removal of seasonal patterns is known as seasonal adjustment. Seasonal effects can be represented using binary or categorical variables, the use of trigonometric functions.

Define stationarity.

Stationarity is the formal mathematical concept corresponding to the stability of a time series of data. A series is said to be (weakly) stationary if

- The mean $E[y_t]$ does not depend on t
- The covariance between y_s and y_t depends only on the difference between time units, $|t-s|$.

Thus, a weakly stationary series has a constant mean as well as a constant variance (homoscedastic). Another type of stationarity known as strict ,or strong, stationarity requires that the entire distribution be constant over time, not just the mean and the variance.

Define white noise.

A white noise process is a stationary process that displays no apparent patterns through time – it is i.i.d., identically and independently dis- tributed.

Define random walk model.

A random walk is the partial sum of a white noise process. The random walk is not a stationary process because the variability, and possibly the mean, depends on the time point at which the series is observed. Let $c_1, ..., c_T$ be T observations from a white noise process. A random walk can be expressed recursively as $y_t = y_{t-1} + c_t$. The mean level and variability of the random walk process are: $E[y_t] = y_0 + t\mu_c$ and $Var[y_t] = t\sigma_c^2$, where $E[c_t] = \mu_c$ and $Var[c_t] = \sigma_c^2$. Hence, as long as there is some variability in the white noise process , the random walk is nonstationary in the variance. Further, if $\mu_c \neq 0$, then the random walk is nonstationary in the mean.

Define the predictive interval for a forecast of the random walk.

To forecast a random walk, we begin with our most recent observation, y_T. We denote the change in y by c_t, so that $c_t = y_t - y_{t-1}$, with average μ_c and standard deviation σ_c. Then an approximate 95% prediction interval for the k-step forecast is $y_{T+k} = k\mu_c \pm 2\sigma_c\sqrt{k}$

Identify random walk versus linear trend time series models.

The linear trend in time model can be written as $y_t = \beta_0 + \beta_1 t + \epsilon_t$, $\{\epsilon_t\}$ is a white noise process. On the other hand, a random walk can be modeled as a partial sum of white noise: decomposing the white noise process into a mean μ_c plus another white noise process $c_t = \mu_c + \epsilon_t$, a random walk model can be written as $y_t = y_0 + \mu_c t + u_t$, where $u_t = \sum_{j=1}^{t} \epsilon_j$. The two models are similar in that the deterministic portion is a linear function of time. The difference is in the error component. The error component for the linear trend in time model is a stationary, white noise process. The error component for the random walk model is nonstationary because it is the partial sum of white noise processes. That is, the error component is also a random walk.

Define filters.

A filter is a procedure for reducing observations to white noise. In regression, we accomplish this by simply subtracting the regression function from the observations. We can reduce a random walk series to a white noise process by taking differences of observations.

List 5 common statistics for comparing forecasts.

1. The mean error statistic measures trends that are not anticipated by the model: $ME = \frac{1}{T}\sum_{t=1}^{T} \hat{\epsilon_t}$
2. The mean percentage error is also a measure of trend, but examines error relative to the actual value: $MPE = \frac{100}{T}\sum_{t=1}^{T} \frac{\hat{\epsilon_t}}{y_t}$
3. The mean square error statistic can detect more than trend patterns: $MSE = \frac{1}{T}\sum_{t=1}^{T} \hat{\epsilon_t}^2$
4. The mean absolute error can detect more than trend patterns. The units of MAE are the same as the dependent variable: $MSE = \frac{1}{T}\sum_{t=1}^{T} |\hat{\epsilon_t}|$

5. The mean absolute percentage error statistic can detect more than trend patterns. It examines error relative to the actual value: $MAPE = \frac{100}{T}\sum_{t=1}^{T}|\frac{\hat{\epsilon}_t}{y_t}|$

3.2 Autocorrelations and Autoregressive Models ([REG] Ch. 8:1-4)

Define lag-k autocorrelation statistics

To summarize the linear relationship between observations that are k time units apart: $r_k = \frac{\sum_{t=k+1}^{T}(y_{t-k}-\overline{y})(y_y=\overline{y})}{\sum_{t=k+1}^{T}(y_{t-k}-\overline{y})^2}$.

Describe AR(1) model and its properties.

The autoregressive model of order 1 , denoted by AR(1), is written as $y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t$ where $\{\epsilon_t\}$ is a white noise process such that $Cov(\epsilon_{t+k}, y_t) = 0$ for $k > 0$.

- β_1 this is restricted to be between -1 and 1. By making this restriction, the AR(1) series $\{y_t\}$ is stationary. Note that if $\beta_1 = 1$, then the model is a random-walk and hence is nonstationary. If $\beta_1 = 0$, then the model reduces to a white noise process.
- The correlation between points k time units apart $\rho^k = \frac{Cov(y_t,y_{t-k})}{\sqrt{Var(y_t)Var(y_{t-k})}} = \frac{Cov(y_t,y_{t-k})}{\sigma_y^2}$. For a (stationary) AR(1) model, $\rho^k = \beta_1^k$.
- Under the hypothesis of no autocorrelation, a good approximation to the standard error of the lag k autocorrelation statistics is $se(r_k) = \frac{1}{\sqrt{T}}$.
- To estimate the variance of the error terms: $s^2 = \frac{1}{T-3}\sum_{t=2}^{T}(e_t - \overline{e})^2$
- The smoothed series (i.e. the values fitted under the model) for the AR(1) model is: $\hat{y}_t = b_0 + b_1 y_{t-1}$.
- The chain rule of forecasting: For an AR(1) model, the k-step ahead forecast is recursively determined by $\hat{y}_{T+k} = b_0 = b_1\hat{y}_{T+k-1}$.
- The k-step ahead forecast interval for an AR (1) model is $\hat{y}_{T+k} \pm$ [t-value]$s\sqrt{1 + b_1^2 + ... + b^{2(k-1)}}$, where t-value is the desired percentile from the t-distribution using $df = T - 3$ degrees of freedom.

3.3 Forecasting and Time Series Models ([REG] Ch. 9:1-5)

Define moving average and exponential smoothing, and their recursive equations

The moving, or running, average estimate is defined by $\hat{s}_t = \frac{y_t+y_{t-1}+...+y_{t-k+1}}{k}$, where k is the running average length. The choice of k depends on the amount of smoothing desired. The larger the value of k, the smoother is the estimate. Expressing the forecast recursively: $\hat{s}_t = \hat{s}_{t-1} + \frac{y_t - y_{t-k}}{k}$

Exponential smoothing estimates are weighted averages of past values of a series, where the weights are given by a series that becomes exponentially small. Let w be a weight number that is between zero and one, and consider the weighted average: $\frac{y_t+wy_{t-1}+w^2y_{t-2}+w^3y_{t-3}+...}{1/(1-w)}$. This is a weighted average because the weights $w_k(1-w)$ sum to one by a geometric series expansion. Expressing the forecast recursively: $\hat{s}_t = \hat{s}_{t-1} + (1-w)(y_t - \hat{s}_{t-1}) = (1-w)y_t + w\hat{s}_{t-1}$

Compare fixed seasonal effects, seasonal autoregression and seasonal exponential smoothing.

For handling seasonal patterns in time series:

- A fixed seasonal effects model represents the seasonal component S_t as a function of time t. The two most important examples are the seasonal binary and trigonometric functions.
- A seasonal autoregressive model of order P represents the correlation between y_t and y_{t-k} using autoregressive models, with P lagged response explanatory variables: $y_t = \beta_0 + \beta_1 y_{t-k} + ... + \beta_P y_{t-Pk}$.
- The Holt-Winter additive seasonal model is a seasonal exponential smoothing method – with three smoothing parameters for the level, trend and seasonality respectively – that appears to work well in practice.

Define the augmented Dickey-Fuller test for unit roots.

This is the t-statistic associated with the y_{t-1} variable using ordinary least squares on the following equation $y_t - y_{t-1} = \beta_0 + (\phi - 1)y_{t-1} + \beta_1 t + \sum_{j=1}^{P} \phi_j (y_{t-j} - y_{t-j-1} + \epsilon_t$ In this equation, we have augmented the possibly serially correlated disturbance term by autoregressive terms in the differences $\{y_{t-j} - y_{t-j-1}\}$. The idea is that these terms serve to capture serial correlation in the disturbance term, where results for a number of choices of lags P should be checked to reach qualitatively similar conclusions. The t-statistic, which does not follow the usual t-distribution (because $\{y_{t-1}\}$ is a random-walk process under the null hypothesis) but rather follows a special Dickey-Fuller distribution, tests $H_0 : \phi = 1$ versus the one-sided alternative that $H_a : \phi < 1$.

Define ARCH and GARCH models.

Many financial time series exhibit volatility clustering , that is, periods of high volatility (large changes in the series) followed by periods of low volatility. We can allow for changing variances by conditioning on the past and still retain a weakly stationary model.

The autoregressive changing heteroscedasticity model of order p, ARCH (p) , is due to Engle (1982). It specifies that the conditional variance is determined recursively by $\sigma_t^2 = w + \gamma_1 \epsilon_{t-1}^2 + ... + \gamma_p \epsilon_{t-p}^2$ where $w > 0$ is the "long-run" volatility parameter and $\gamma_1, ..., \gamma_p$ are coefficients such that $\gamma_j \geq 0$ and $\sum_{j=1}^{p} \gamma_j = 1$. Engle provided additional mild conditions to ensure that $\{\epsilon_t\}$ is weakly stationarity. Thus, despite having a changing conditional variance, the unconditional variance remains constant over time

The generalized ARCH model of order p, GARCH(p,q), complements the ARCH model in the same way that the moving average complements the autoregressive model. The conditional variance is determined recursively by $\sigma_t^2 - \delta_1 \sigma_{t-1}^2 - ... - \delta_q \sigma_{t-q}^2 = w + \gamma_1 \epsilon_{t-1}^2 + ... + \gamma_p \epsilon_{t-p}^2$. In addition to the ARCH(p) requirements, we also need $\delta_j \geq 0$ and $\sum_{j=1}^{p} \gamma_j + \sum_{j=1}^{q} \delta_j < 1$. The GARCH(p,q) is also a weakly stationary model, with mean zero and (unconditional) variance $Var(\epsilon_t) = w/(1 - \sum_{j=1}^{p} \gamma_j - \sum_{j=1}^{q} \delta_j)$.

4. Topic: Principal Components Analysis

Learning Objectives: The Candidate will understand key concepts concerning principal components analysis.

1. Define principal components.
2. Interpret the results of a principal components analysis, considering loading factors and proportion of variance explained.
3. Explain uses of principal components.

4.1 Principal component analysis ([ISL] Ch. 10.2)

Define principal component analysis.
The process by which principal components are computed, and the subsequent use of these components in understanding the data. PCA is an unsupervised approach, since it involves only a set of features. Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

Describe how principal components are found.
PCA finds a low-dimensional representation of a data set that contains as much as possible of the variation. PCA seeks a small number of dimensions that are as interesting as possible, where the concept of interesting is measured by the amount that the observations vary along each dimension. Each of the dimensions found by PCA is a linear combination of the p features. The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features $Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$ that has the largest variance. In other words, the first principal component loading vector solves the optimization problem: $\max_{\phi_{11},...,\phi_{p1}}\{\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{j1}x_{ij})^2\}$ subject to $\sum_{j=1}^{p}\phi_{j1}^2 = 1$. We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance. We can then iteratively find the second principal component Z_2 (the linear combination of features that has maximal variance out of all linear combinations that are uncorrelated with previously found components), third component Z_3, and so on.

Define loadings of a principal component.
Coefficients $\phi_{11}, \phi_{21}, ..., \phi_{p1}$ in the normalized linear combination of the features to form principal components of the set of features.

Describe another interpretation of the principal component.
Principal components provide low-dimensional linear surfaces that are closest to the observations. Using this interpretation, together the first M principal component score vectors and the first M principal

component loading vectors provide the best M-dimensional approximation (in terms of Euclidean distance) to the ith observation: $x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}$

Describe the effect of scaling on the principal component.

The results obtained when we perform PCA will also depend on whether the variables have been individually scaled (each multiplied by a different constant). If we perform PCA on the unscaled variables, then the first principal component loading vector will have large loadings on the variables with the highest variance. However, this result is simply a consequence of the scales or units on which the variables were measured.

Describe uniqueness of the principal component.

Each principal component loading vector is unique, up to a sign flip. The signs may differ because each principal component loading vector specifies a direction in p-dimensional space: flipping the sign has no effect as the direction does not change.

Define proportion of variance explained and scree plot.

The proportion of variance explained (PVE) of the mth principal component is given by the ratio the variance explained by the mth principal component, and the total variance present in a data set (assuming that the variables have been centered to have mean zero): $\text{PVE} = \frac{\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{jm}x_{ij})^2}{\sum_{j=1}^{p}\sum_{i=1}^{n}x_{ij}^2}$. A scree plot depicts the proportion of variance explained by each of the principal components.

Describe how number of principal components is decided.

We typically decide on the number of principal components required to visualize the data by examining a scree plot. We choose the smallest number of principal components that are required in order to explain a sizable amount of the variation in the data. This is done by eyeballing the scree plot, and looking for a point at which the proportion of variance explained by each subsequent principal component drops off. This is often referred to as an elbow in the scree plot. This type of visual analysis is inherently ad hoc. Unfortunately, there is no well-accepted objective way to decide how many are enough.

4.2 Lab: Principal Components ([ISL] Ch 10.4, 10.6.1)


```r
states=row.names(USArrests)
states
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

```r
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

```r
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```r
apply(USArrests, 2, var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

```r
pr.out=prcomp(USArrests, scale=TRUE)
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
pr.out$center
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```r
pr.out$scale
```

```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

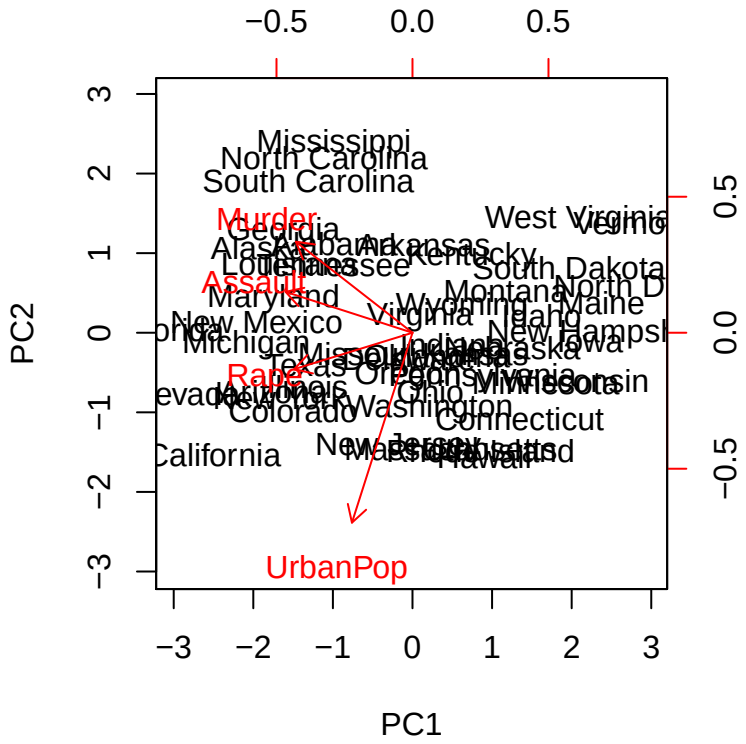
```r
pr.out$rotation
```

```
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

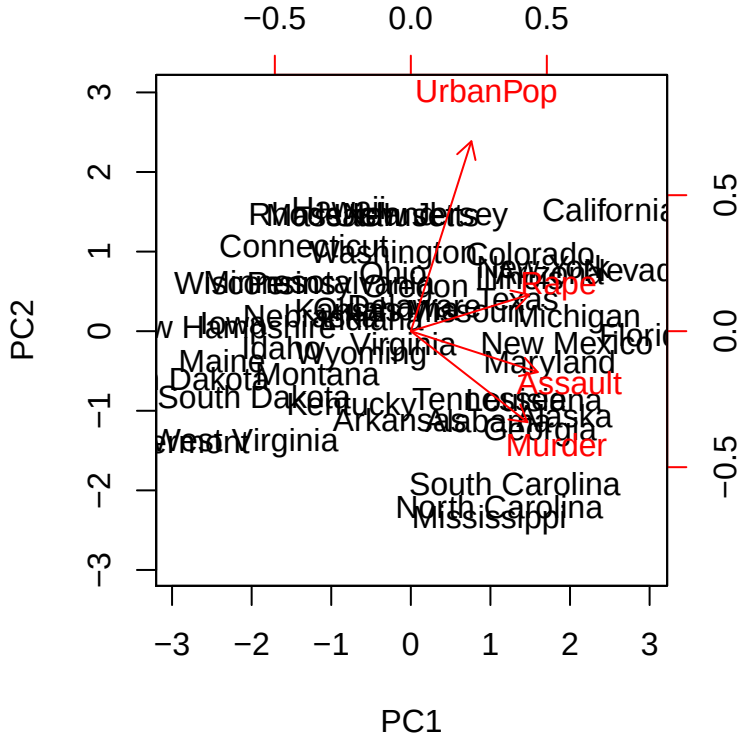
```r
dim(pr.out$x)
```

```
## [1] 50  4
```

```r
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
```

```
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
```

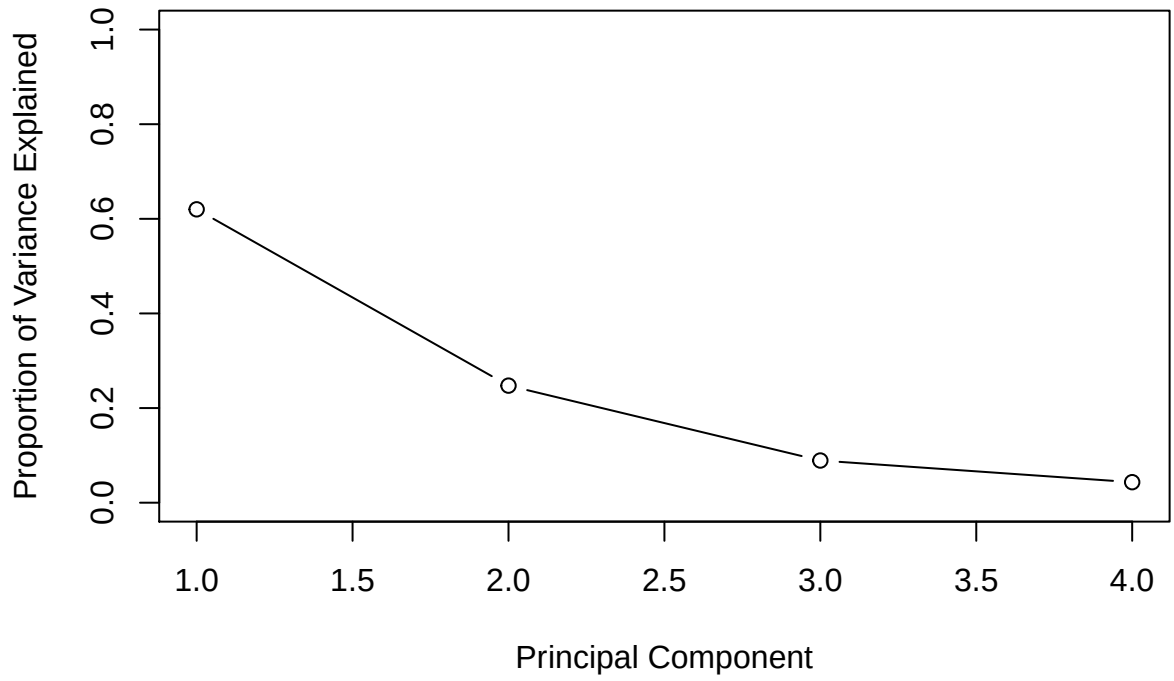
```
pr.var=pr.out$sdev^2
pr.var
```

```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

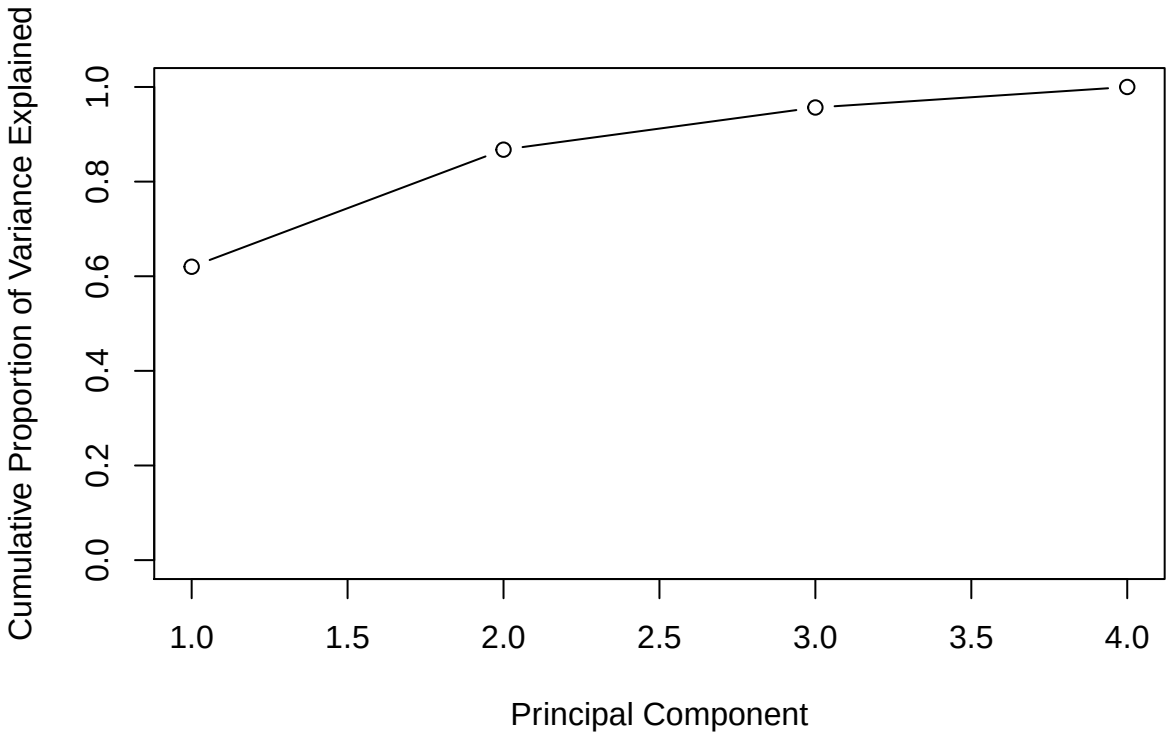
```
pve=pr.var/sum(pr.var)
pve
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

```
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=
```




```
a=c(1,2,8,-3)
cumsum(a)
```

```
## [1]  1  3 11  8
```

```
# The NCI60 data

library(ISLR)
nci.labs=NCI60$labs
nci.data=NCI60$data
dim(nci.data)
```

```
## [1]   64 6830
```

```
nci.labs[1:4]
```

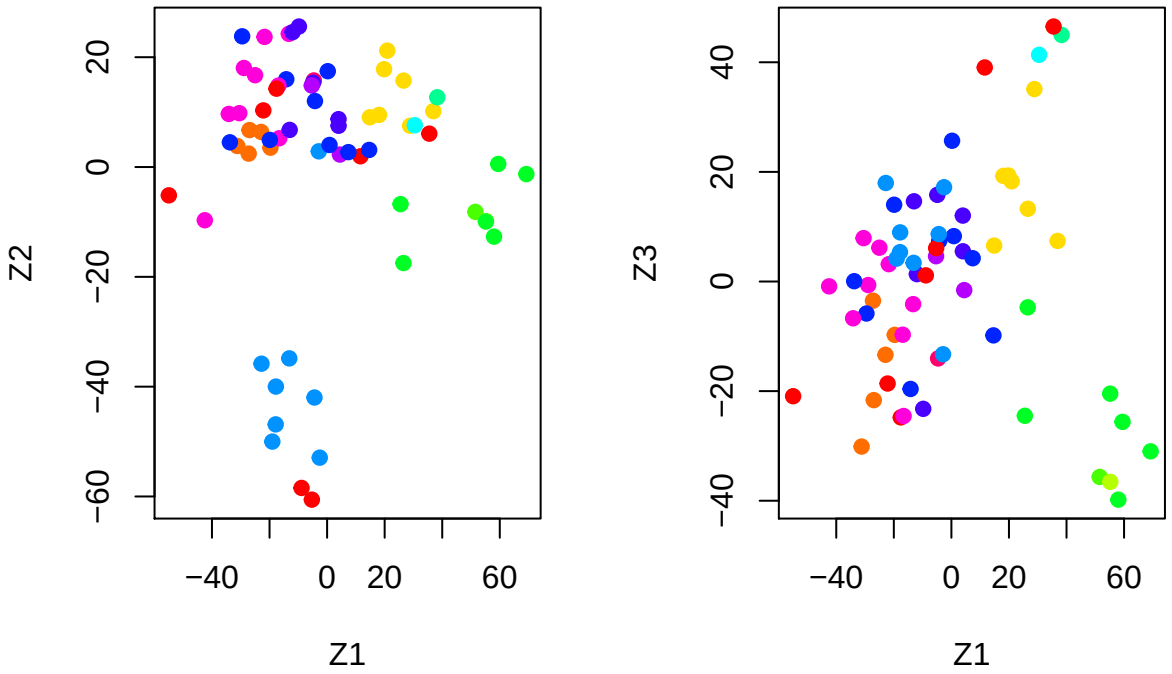
```
## [1] "CNS"   "CNS"   "CNS"   "RENAL"
```

```
table(nci.labs)
```

```
## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

```
# 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")
```

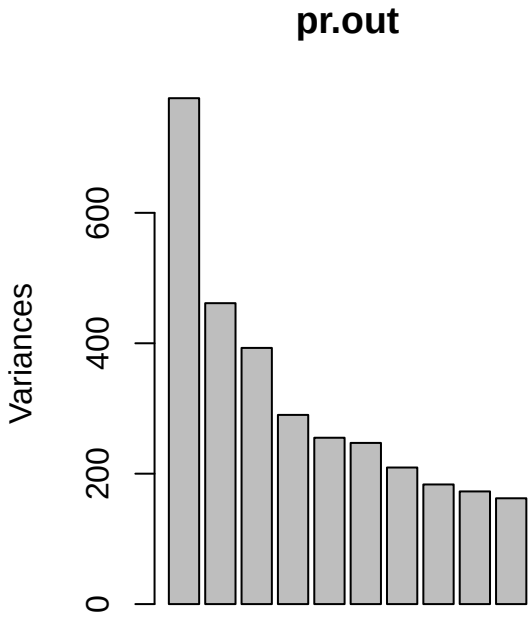



```
summary(pr.out)
```

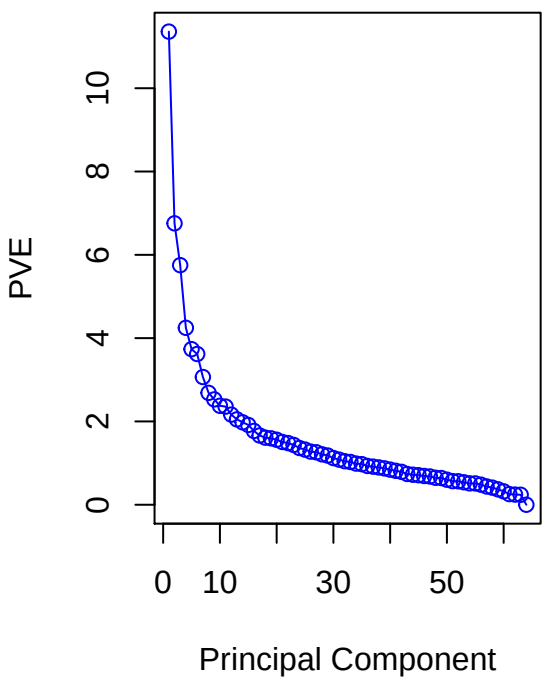
```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     27.8535 21.48136 19.82046 17.03256 15.97181
## Proportion of Variance  0.1136  0.06756  0.05752  0.04248  0.03735
## Cumulative Proportion   0.1136  0.18115  0.23867  0.28115  0.31850
##                             PC6      PC7      PC8      PC9     PC10
## Standard deviation     15.72108 14.47145 13.54427 13.14400 12.73860
## Proportion of Variance  0.03619  0.03066  0.02686  0.02529  0.02376
## Cumulative Proportion   0.35468  0.38534  0.41220  0.43750  0.46126
##                            PC11     PC12     PC13     PC14     PC15
## Standard deviation     12.68672 12.15769 11.83019 11.62554 11.43779
## Proportion of Variance  0.02357  0.02164  0.02049  0.01979  0.01915
## Cumulative Proportion   0.48482  0.50646  0.52695  0.54674  0.56590
##                            PC16     PC17     PC18     PC19    PC20
## Standard deviation     11.00051 10.65666 10.48880 10.43518 10.3219
## Proportion of Variance  0.01772  0.01663  0.01611  0.01594  0.0156
## Cumulative Proportion   0.58361  0.60024  0.61635  0.63229  0.6479
##                            PC21    PC22    PC23    PC24    PC25    PC26
## Standard deviation     10.14608 10.0544 9.90265 9.64766 9.50764 9.33253
## Proportion of Variance  0.01507  0.0148 0.01436 0.01363 0.01324 0.01275
## Cumulative Proportion   0.66296  0.6778 0.69212 0.70575 0.71899 0.73174
##                           PC27   PC28    PC29    PC30    PC31    PC32
## Standard deviation     9.27320 9.0900 8.98117 8.75003 8.59962 8.44738
## Proportion of Variance 0.01259 0.0121 0.01181 0.01121 0.01083 0.01045
## Cumulative Proportion  0.74433 0.7564 0.76824 0.77945 0.79027 0.80072
##                           PC33    PC34    PC35    PC36    PC37    PC38
```

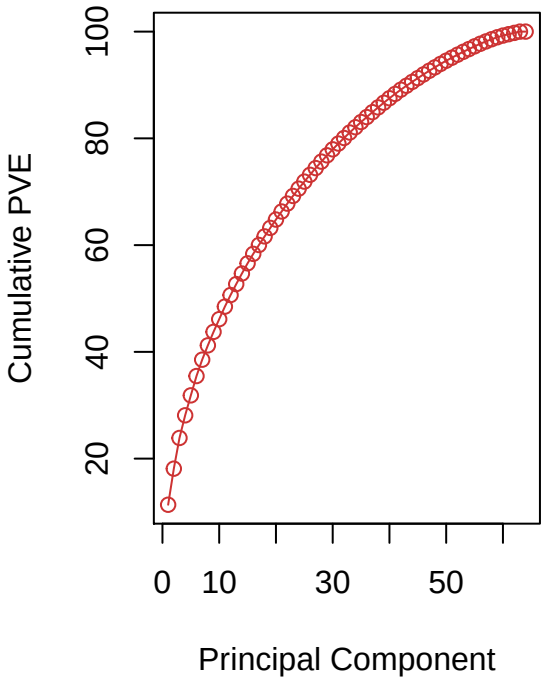
```
## Standard deviation     8.37305 8.21579 8.15731 7.97465 7.90446 7.82127
## Proportion of Variance 0.01026 0.00988 0.00974 0.00931 0.00915 0.00896
## Cumulative Proportion  0.81099 0.82087 0.83061 0.83992 0.84907 0.85803
##                           PC39    PC40    PC41   PC42    PC43   PC44
## Standard deviation     7.72156 7.58603 7.45619 7.3444 7.10449 7.0131
## Proportion of Variance 0.00873 0.00843 0.00814 0.0079 0.00739 0.0072
## Cumulative Proportion  0.86676 0.87518 0.88332 0.8912 0.89861 0.9058
##                           PC45   PC46    PC47    PC48    PC49    PC50
## Standard deviation     6.95839 6.8663 6.80744 6.64763 6.61607 6.40793
## Proportion of Variance 0.00709 0.0069 0.00678 0.00647 0.00641 0.00601
## Cumulative Proportion  0.91290 0.9198 0.92659 0.93306 0.93947 0.94548
##                           PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     6.21984 6.20326 6.06706 5.91805 5.91233 5.73539
## Proportion of Variance 0.00566 0.00563 0.00539 0.00513 0.00512 0.00482
## Cumulative Proportion  0.95114 0.95678 0.96216 0.96729 0.97241 0.97723
##                           PC57   PC58    PC59    PC60    PC61    PC62
## Standard deviation     5.47261 5.2921 5.02117 4.68398 4.17567 4.08212
## Proportion of Variance 0.00438 0.0041 0.00369 0.00321 0.00255 0.00244
## Cumulative Proportion  0.98161 0.9857 0.98940 0.99262 0.99517 0.99761
##                           PC63      PC64
## Standard deviation     4.04124 1.237e-14
## Proportion of Variance 0.00239 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

```
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")
```


PVE
0
2
4
6
8
10
0
10
30
50
Principal Component


Cumulative PVE
20
40
60
80
100
0
10
30
50
Principal Component

5. Topic: Decision Trees

Learning Objectives: The Candidate will understand key concepts concerning decision tree models.

1. Explain the purpose and uses of decision trees.
2. Explain and interpret decision trees, considering regression trees and recursive binary splitting.
3. Explain and interpret bagging, boosting, and random forests.
4. Explain and interpret classification trees, their construction, Gini index, and entropy.
5. Compare decision trees to linear models.
6. Interpret the results of a decision tree analysis.

5.1 The Basics of Decision Trees ([ISL] Ch. 8.1)

Define tree-based or decision tree methods

These involve stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs. The set of splitting rules used to segment the predictor space can be summarized in a tree.

Define leaf, terminal node, internal node and branch of a decision tree

A tree consistes of a series of aplitting rules, that segments observations into players into regions of predictor space. These regions are known as terminal nodes or leaves of the tree. Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree. The points along the tree where the predictor space is split are referred to as internal nodes. The segments of the trees that connect the nodes are branches.

Explain recursive binary splitting

A top-down, greedy approach to construct decision trees. It begins at the top of the tree (at which point all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree. At each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step. We first select that pre- dictor and the cutpoint s such that splitting the predictor space into the regions leads to the greatest possible reduction in a cost function (RSS). Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data in one of the two previously identified regions further so as to minimize the cost within each of the resulting regions. The process continues until a stopping criterion is reached.

Explain cost complexity pruning

A better strategy to reduce the complexity of a tree (that is less likely to overfit the data) is to grow a very large tree, and then prune it back in order to obtain a subtree. However, estimating the test error from cross-validation error for every possible subtree is too cumbersome, since there is an extremely large number of possible subtrees. Cost complexity pruning (also known as weakest link pruning) considers a sequence of trees indexed by a nonnegative tuning parameter α, rather than considering every possible subtree. For each value of α there corresponds a subtree T such that $\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$ is as small as possible, where $|T|$ indicates the number of terminal nodes of the tree T, R_m is the subset of predictor space corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the predicted response associated with R_m. The tuning parameter α controls a trade-off between the subtree's complexity (number of terminal nodes) and its fit to the training data.

List the steps of building a regression tree with cost-complexity pruning

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, ..., K$:
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α. Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Define Gini index and entropy for classification trees

Alternate criteria for making the binary splits in a classification tree, as the simple classification error rate (i.e. the fraction of the training observations in that region that do not belong to the most common class) is not sufficiently sensitive for tree-growing. The Gini index, a measure of total variance across the K classes, is defined by $G = \sum_{k=1}^{K} \hat{\rho}_{mk}(1 - \hat{\rho}_{mk})$, where $\hat{\rho}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. It is not hard to see that the Gini index takes on a small value if all of the $\hat{\rho}_{mk}$'s are close to zero or one. For this reason the Gini index is referred to as a measure of node purity – a small value indicates that a node contains predominantly observations from a single class. Entropy, given by $D = \sum_{k=1}^{K} \hat{\rho}_{mk} \log \hat{\rho}_{mk}$. The entropy will take on a value near zero if the $\hat{\rho}_{mk}$'s are all near zero or near one, when the mth node is pure.

Compare decision trees to linear models

If there is a highly non-linear and complex relationship between the features and the response, then decision trees (which assume a model of the form $f(x) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$) may outperform linear regression models (which assume the form $f(x) = \beta_0 \sum_{j=1}^{p} X_j \beta_j$)

List Advantages and Disadvantages of Trees

Advantages

1. Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
2. Decision trees may more closely mirror human decision-making than do the regression and classification approaches.

3. Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if the trees are small).
4. Trees can easily handle qualitative predictors without the need to create dummy variables

Disadvantages:

1. Trees generally do not have the same level of predictive accuracy as some of the other regression and classification
2. Trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

By aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved.

5.2 Bagging, Random Forests, Boosting ([ISL] Ch. 8.2)

Define bagging for regression trees and classification trees

Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method. Although we generally do not have access to multiple training sets. we can bootstrap, by taking repeated samples from the (single) training data set. We simply construct B regression trees using B different bootstrapped training sets, and average the resulting predictions. These trees are grown deep and are not pruned. Hence each individual tree has has high variance, but low bias. Averaging these trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure. Bagging be extended to a classification problem by approaches such as a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

Describe the out-of-bag approach for estimating test error

The remaining observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. We can predict the response for the ith observation using each of the trees in which that observation was OOB. In order to obtain a single prediction for the ith observation, we can average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal). An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB MSE (for a regression problem) or classification error (for a classification problem) can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

Explain variable importance in bagged trees

Unlike a single tree, then we bag a large number of trees, it is no longer clear which variables are most important to the procedure. One can obtain an overall summary of variable importance by recording the total amout of error (e.g. RSS for regression trees or the Gini index for classification trees) is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.

Explain random forests

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. When building decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors: the split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose

$m \approx \sqrt{p}$. In the presence of one strong predictor, this gives other predictors more of a chance; otherwise, most or all of the trees will use this strong predictor in the top split, and the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance. Hence, we can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

Explain boosting

Boosting also fits a separate decision tree to each copy, and then combine all of the trees in order to create a single predictive model. Except that the trees are grown sequentially (each tree is grown using information from previously grown trees) and does not involve bootstrap sampling (each tree is fit on a modified version of the original data set). The boosting approach instead learns slowly. Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals. Each of these trees can be rather small, with just a few terminal nodes,

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, ..., B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree: $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
   (c) Update the residuals, $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
3. Output the boosted model, $\hat{f}^b(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$

Boosting has three tuning parameters:

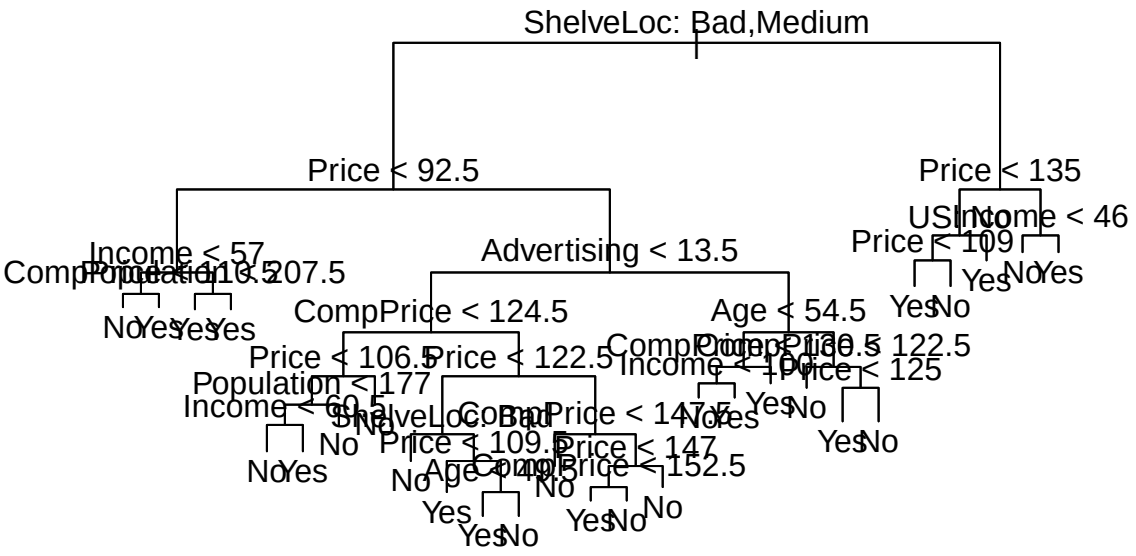
- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem.
- The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

5.3 Lab: Classification ([ISL] Ch 8.3)


```
# 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)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.carseats)
text(tree.carseats,pretty=0)
```




```
tree.carseats
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 400 541.500 No ( 0.59000 0.41000 )
##     2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
##       4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
##         8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
##          16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
##          17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
##         9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
##          18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
##          19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
```

```
##       5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
##        10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
##          20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
##            40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
##              80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
##               160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
##               161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
##              81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
##            41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
##          21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
##            42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
##              84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
##              85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
##               170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
##               171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
##                 342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
##                 343) Age > 49.5 11   6.702 No ( 0.90909 0.09091 ) *
##            43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
##              86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
##              87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
##               174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
##                 348) CompPrice < 152.5 7   5.742 Yes ( 0.14286 0.85714 ) *
##                 349) CompPrice > 152.5 5   5.004 No ( 0.80000 0.20000 ) *
##               175) Price > 147 7   0.000 No ( 1.00000 0.00000 ) *
##        11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
##          22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
##            44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
##              88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
##              89) Income > 100 5   0.000 Yes ( 0.00000 1.00000 ) *
##            45) CompPrice > 130.5 11   0.000 Yes ( 0.00000 1.00000 ) *
##          23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
##            46) CompPrice < 122.5 10   0.000 No ( 1.00000 0.00000 ) *
##            47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
##              94) Price < 125 5   0.000 Yes ( 0.00000 1.00000 ) *
##              95) Price > 125 5   0.000 No ( 1.00000 0.00000 ) *
##     3) ShelveLoc: Good 85  90.330 Yes ( 0.22353 0.77647 )
##       6) Price < 135 68  49.260 Yes ( 0.11765 0.88235 )
##        12) US: No 17  22.070 Yes ( 0.35294 0.64706 )
##          24) Price < 109 8   0.000 Yes ( 0.00000 1.00000 ) *
##          25) Price > 109 9  11.460 No ( 0.66667 0.33333 ) *
##        13) US: Yes 51  16.880 Yes ( 0.03922 0.96078 ) *
##       7) Price > 135 17  22.070 No ( 0.64706 0.35294 )
##        14) Income < 46 6   0.000 No ( 1.00000 0.00000 ) *
##        15) Income > 46 11  15.160 Yes ( 0.45455 0.54545 ) *
```

```
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)
```

```
##          High.test
## tree.pred  No Yes
```

```
##       No  104  33
##       Yes  13  50
```

```
(86+57)/200
```

```
## [1] 0.715
```

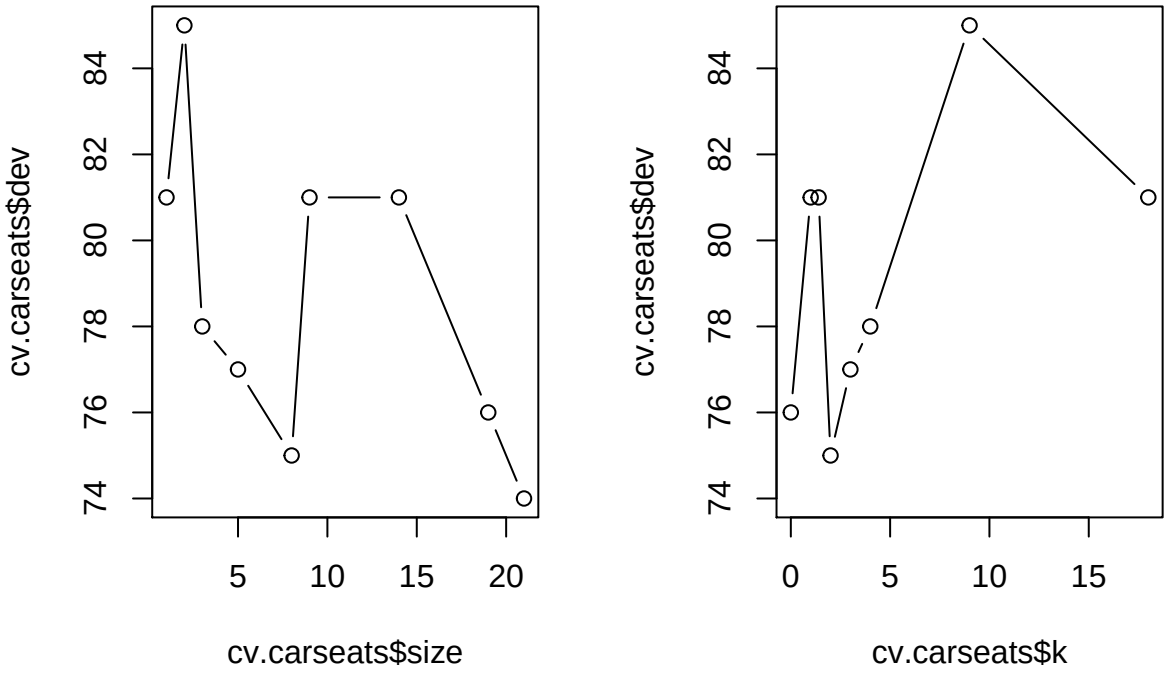
```
set.seed(3)
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
names(cv.carseats)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.carseats
```

```
## $size
## [1] 21 19 14  9  8  5  3  2  1
##
## $dev
## [1] 74 76 81 81 75 77 78 85 81
##
## $k
## [1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
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)
```

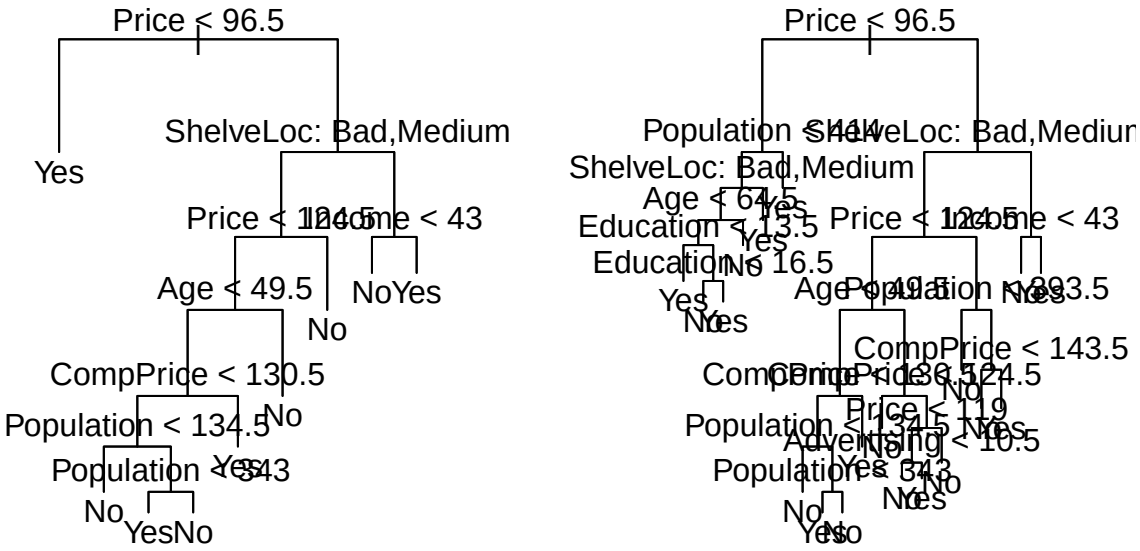
```
text(prune.carseats,pretty=0)
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  97  25
##       Yes 20  58
```

```
(94+60)/200
```

```
## [1] 0.77
```

```
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)
```

```
##          High.test
## tree.pred  No Yes
##       No  102  30
##       Yes  15  53
```

```
(86+62)/200
```

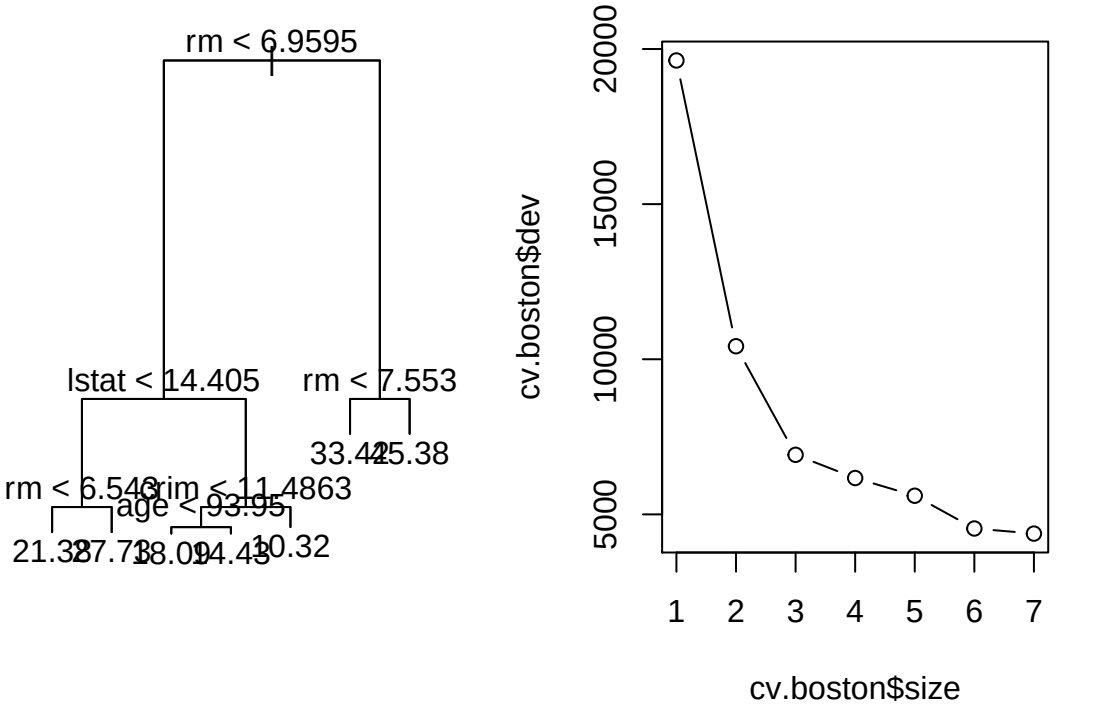
```
## [1] 0.74
```

```
# 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)
```

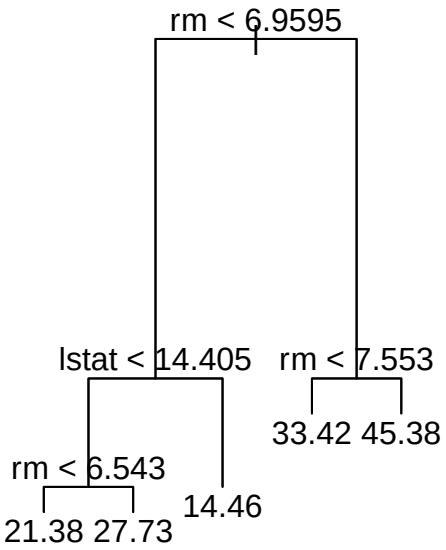
```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
```

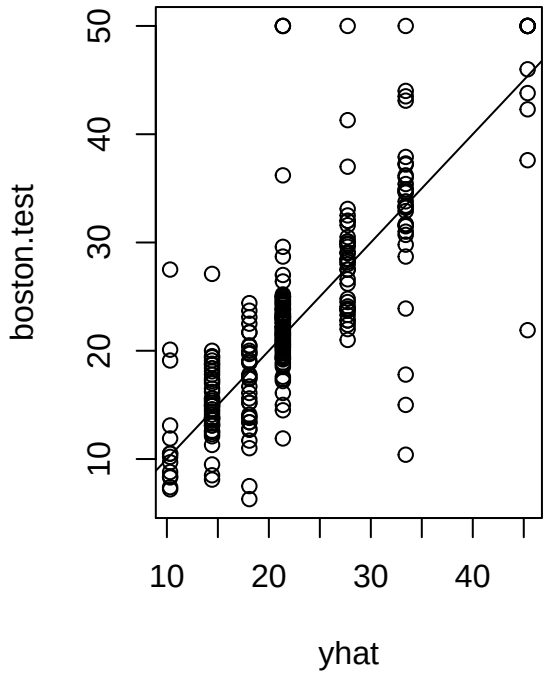
```
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

```
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)
```

```
## [1] 35.28688
```

```
# Bagging and Random Forests

library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE)
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.39601
##                     % Var explained: 85.17
```

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
mean((yhat.bag-boston.test)^2)
```

```
## [1] 23.59273
```

```
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
```

```r
mean((yhat.bag-boston.test)^2)
```

```
## [1] 23.66716
```

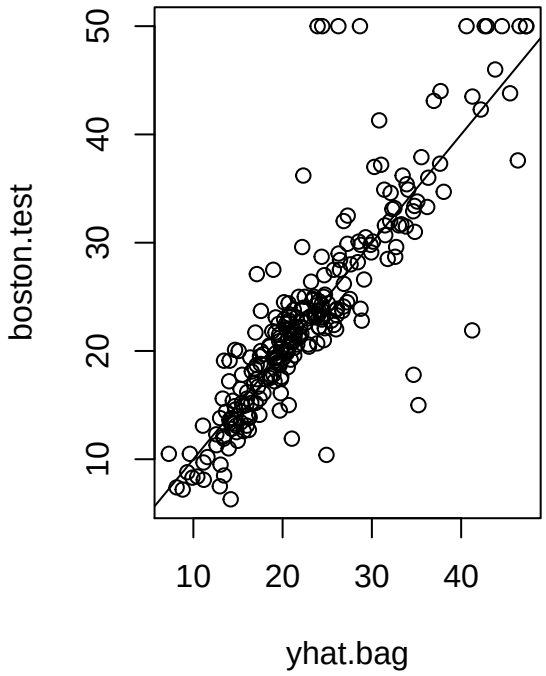
```r
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
```

```
## [1] 19.62021
```

```r
importance(rf.boston)
```

```
##           %IncMSE IncNodePurity
## crim    16.697017    1076.08786
## zn       3.625784      88.35342
## indus    4.968621     609.53356
## chas     1.061432      52.21793
## nox     13.518179     709.87339
## rm      32.343305    7857.65451
## age     13.272498     612.21424
## dis      9.032477     714.94674
## rad      2.878434      95.80598
## tax      9.118801     364.92479
## ptratio  8.467062     823.93341
## black    7.579482     275.62272
## lstat   27.129817    6027.63740
```

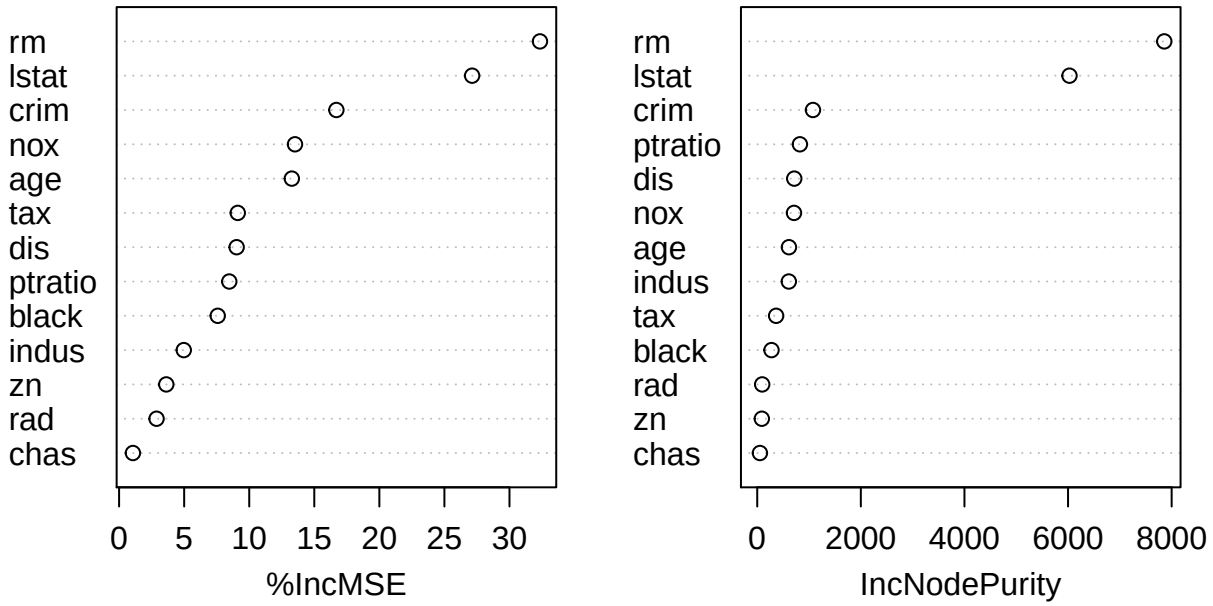
```r
varImpPlot(rf.boston)
```




```r
# Boosting

library(gbm)
```

```
## Loaded gbm 2.1.5
```

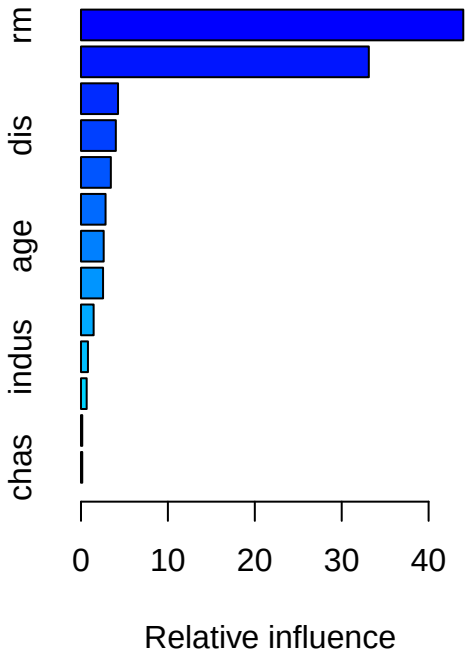
rf.boston



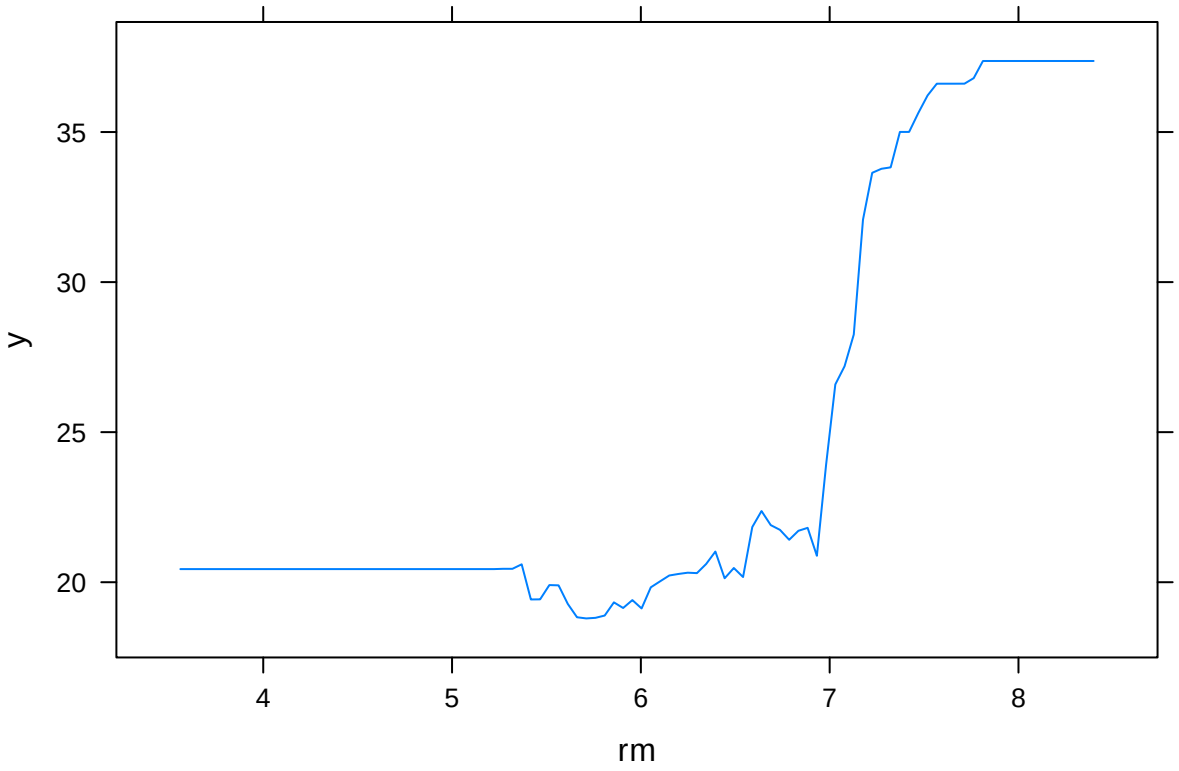

```
set.seed(1)
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4)
summary(boost.boston)
```

```
##             var    rel.inf
## rm           rm 43.9919329
## lstat     lstat 33.1216941
## crim       crim  4.2604167
## dis         dis  4.0111090
## nox         nox  3.4353017
## black     black  2.8267554
## age         age  2.6113938
## ptratio ptratio  2.5403035
## tax         tax  1.4565654
## indus     indus  0.8008740
## rad         rad  0.6546400
## zn           zn  0.1446149
## chas       chas  0.1443986
```

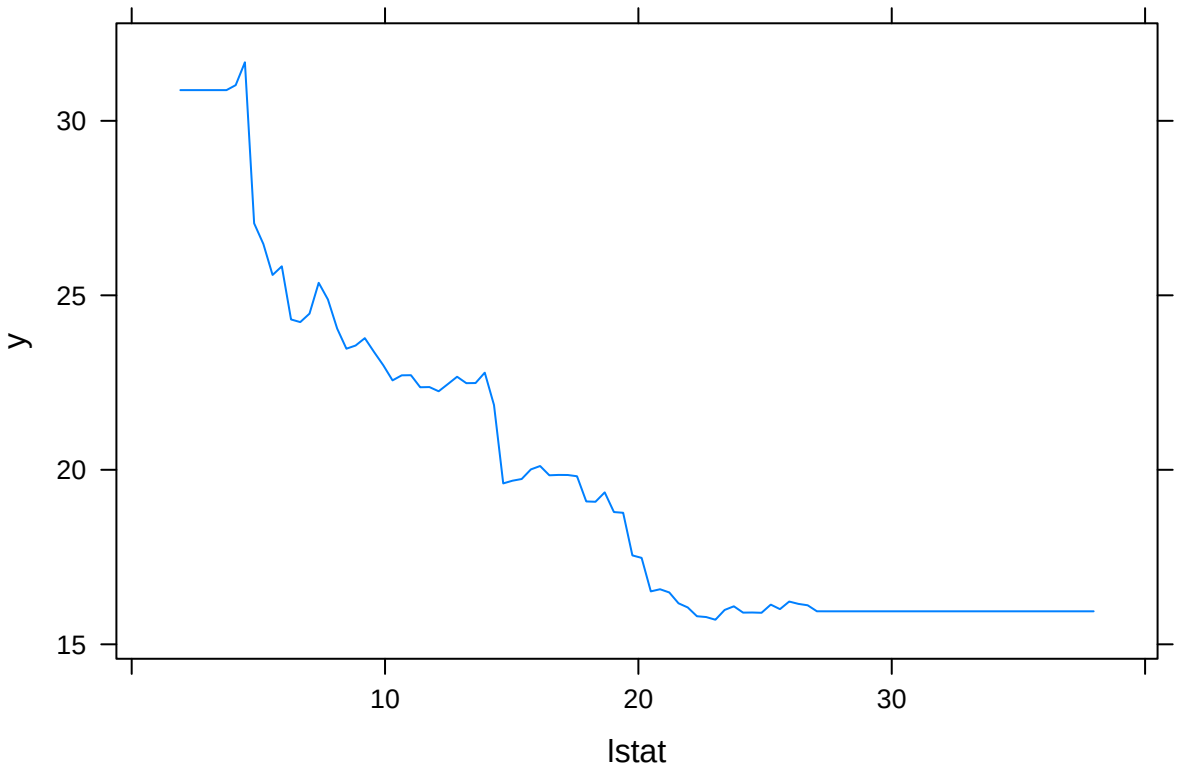
```
par(mfrow=c(1,2))
```




```
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)
```

```
## [1] 18.84709
```

```
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interaction.depth=4,sh
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 18.33455
```

6. Topic: Cluster Analysis

Learning Objectives: The Candidate will understand key concepts concerning cluster analysis.

1. Explain the uses of clustering.
2. Explain K-means clustering.
3. Explain hierarchical clustering.
4. Explain methods for deciding the number of clusters.
5. Compare hierarchical with K-means clustering.

6.1 Clustering methods ([ISL] Ch. 10.3)

Describe the algorithm for k-means clustering.

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

Define bottom-up agglomerative clustering.

The most common type of hierarchical clustering, which is an alternative approach (to K-means clustering) that does not require that we commit to a particular choice of K. bottom-up agglomerative clustering efers to the fact that a dendrogram is built starting from the leaves (at the bottom) and combining clusters up to the trunk.

Describe how to interpret a dendogram.

A dendrogram is generally depicted as an upside-down tree. Each leaf of the dendrogram represents one of the observations. As we move up the tree, some leaves begin to fuse into branches. These correspond to observations that are similar to each other. As we move higher up the tree, branches themselves fuse, either with leaves or other branches. The earlier (lower in the tree) fusions occur, the more similar the groups of observations are to each other. On the other hand, observations that fuse later (near the top of the tree) can be quite different. The height of this fusion, as measured on the vertical axis, indicates how different the two observations are. We draw conclusions about the similarity of two observations based on the location on the vertical axis where branches containing those two observations first are fused.

Define linkage as well as the four types of linkage.

Linkage extends the concept of dissimilarity between a pair of observations to a pair of groups of observations:

1. Complete: Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities.
2. Single: Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
3. Average: Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.
4. Centroid: Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions.

Define inversion.

When two clusters are fused at a height below either of the individual clusters in the dendrogram. Centroid linkage, often used in genomics, suffers from a major drawback in that an inversion can occur. This can lead to difficulties in visualization as well as in interpretation of the dendrogram.

Describe the choice of dissimilarity measure for clustering.

The choice of dissimilarity measure is very important, as it has a strong effect on the resulting dendrogram. Correlation-based distance considers two observations to be similar if their features are highly correlated, even though the observed values may be far apart in terms of Euclidean distance. Correlation-based distance focuses on the shapes of observation profiles rather than their magnitudes. One must also consider whether or not the variables should be scaled to have standard deviation one before the dissimilarity between the observations is computed.

Describe the practical issues with clustering.

1. Small Decisions with Big Consequences: In order to perform clustering, some decisions must be made. In practice, we try several different choices, and look for the one with the most useful or interpretable solution:
   - Should the observations or features first be standardized in some way?
   - In the case of hierarchical clustering, what dissimilarity and type of linkage should be used; Where should we cut the dendrogram?
   - In the case of K-means clustering, how many clusters should we look for in the data?
2. Validating the Clusters Obtained: Whether the clusters that have been found represent true subgroups in the data, or whether they are simply a result of clustering the noise.
3. Other Considerations in Clustering: Clustering methods generally are not very robust to perturbations to the data, a set of clusters formed after removing a random subset of observations can be quite dis-similar. Clustering methods that force every observation into a cluster find clusters that may be heavily distorted due to the presence of outliers that do not belong to any cluster – other approaches like mixture models may be more attractive.
4. A Tempered Approach to Interpreting: Results should not be taken as the absolute truth about a data set but as a starting point for further study. We recommend performing clustering with different

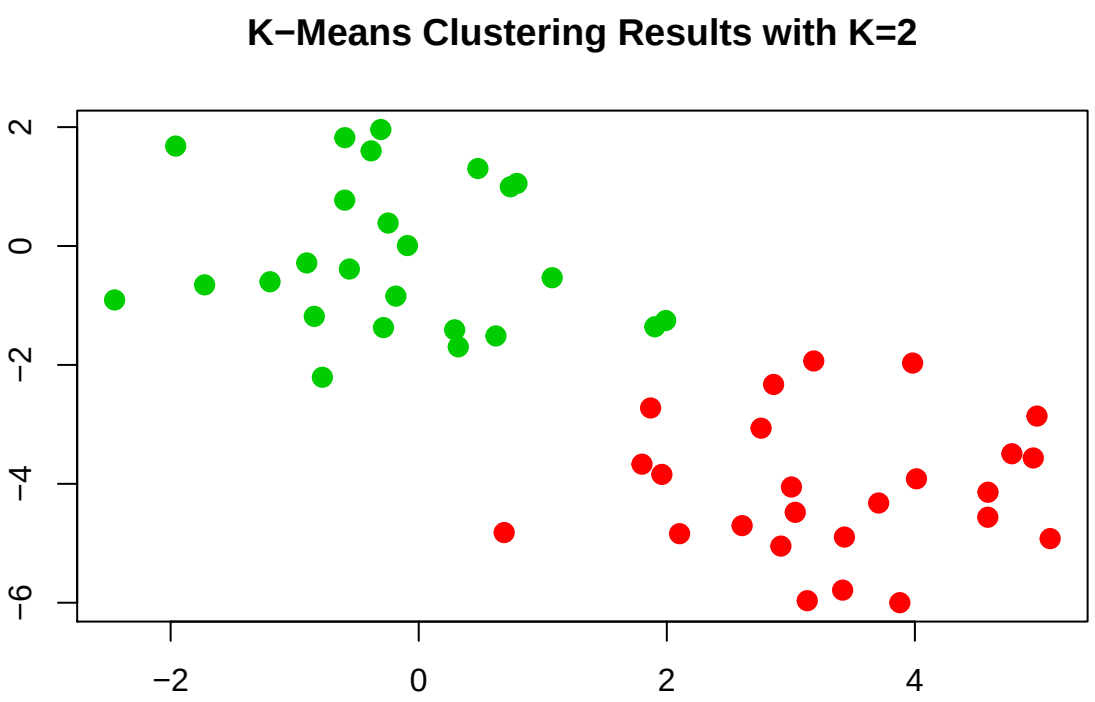
choices of parameters, and clustering subsets of the data in order to get a sense of robustness.

6.2 Lab: Clustering ([ISL] Ch 10.5, 10.6.2)


```r
# 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
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results with K=2", xlab="", ylab="", pch=20, c
```

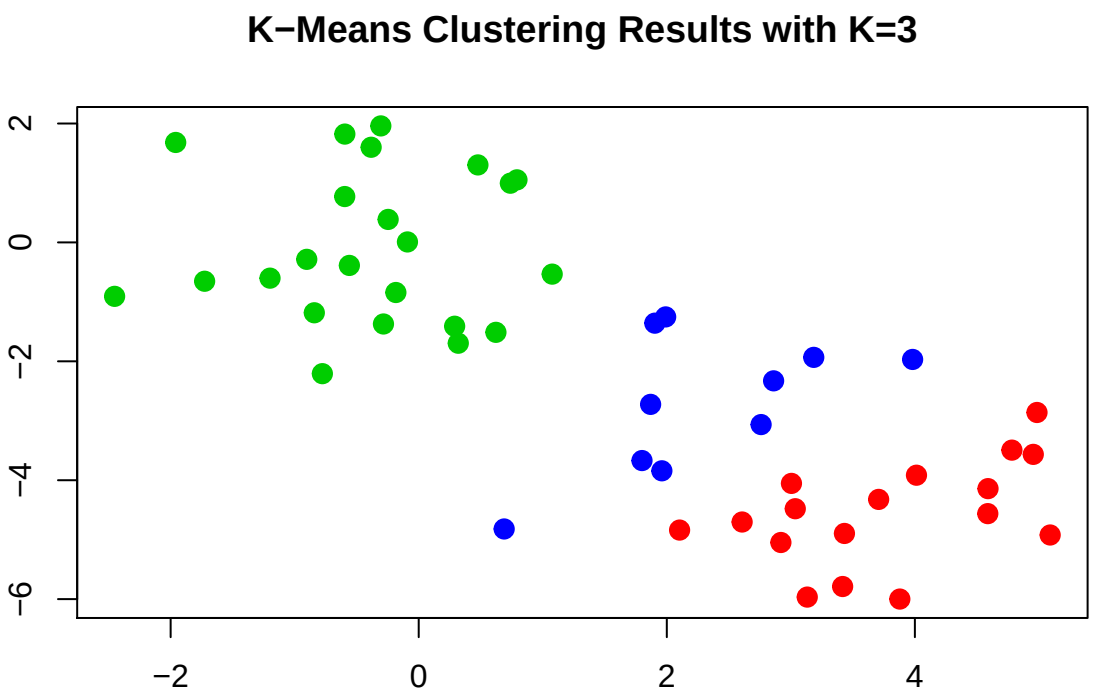



```r
set.seed(4)
km.out=kmeans(x,3,nstart=20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 17, 23, 10
##
## Cluster means:
##         [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2 -0.3820397 -0.08740753
## 3  2.3001545 -2.69622023
##
## Clustering vector:
##  [1] 1 3 1 3 1 1 1 3 1 3 1 3 1 3 1 3 1 1 1 1 1 3 1 1 1 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 25.74089 52.67700 19.56137
##  (between_SS / total_SS =  79.3 %)
```

```
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results with K=3", xlab="", ylab="", pch=20, c
```



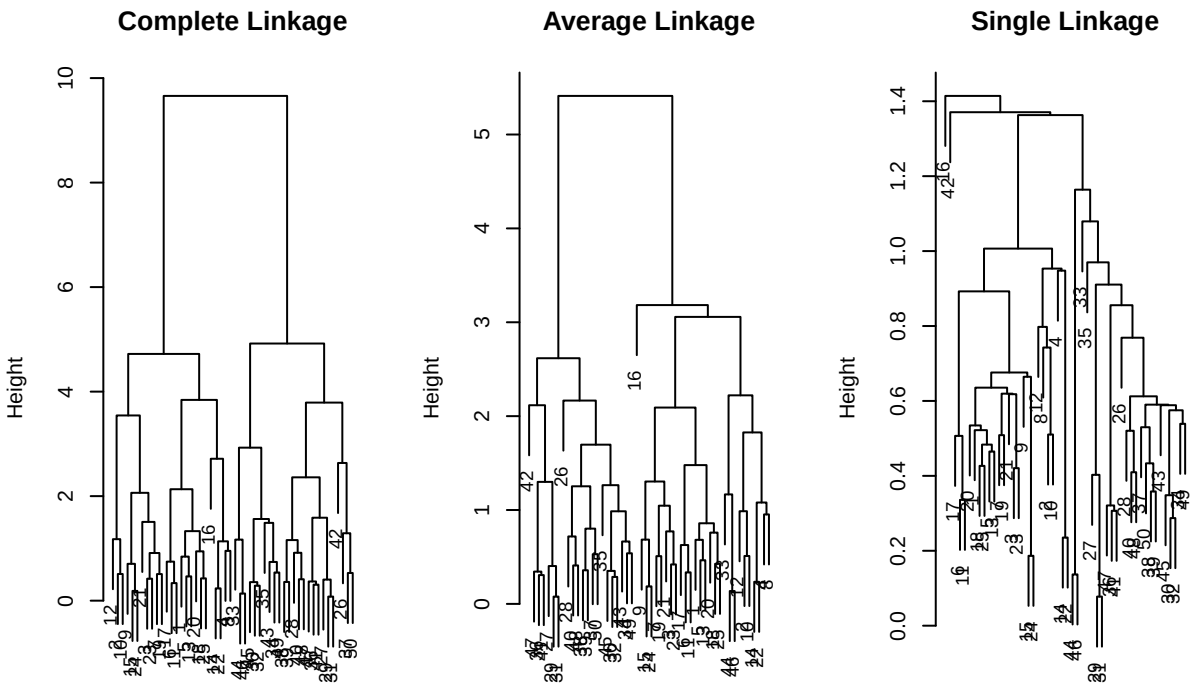

```
set.seed(3)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
```

```
## [1] 97.97927
```

```
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss
```

```
## [1] 97.97927
```

```
# 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)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc.average, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2
## [36] 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
```

```
cutree(hc.single, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
cutree(hc.single, 4)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3
## [36] 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3
```

```
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="", su

# The NCI60 data

library(ISLR)
nci.labs=NCI60$labs
nci.data=NCI60$data
dim(nci.data)
```

```
## [1]   64 6830
```

```
nci.labs[1:4]
```

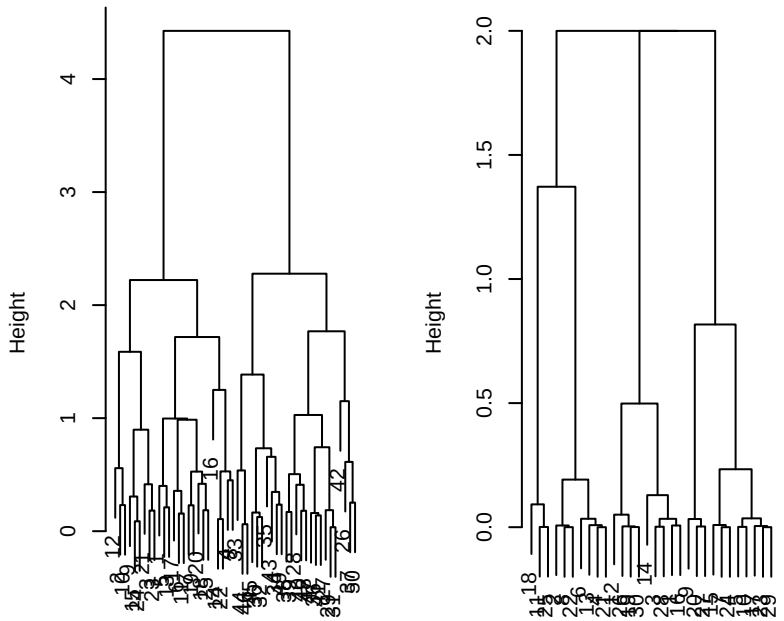
```
## [1] "CNS"   "CNS"   "CNS"   "RENAL"
```

```
table(nci.labs)
```

```
## nci.labs
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

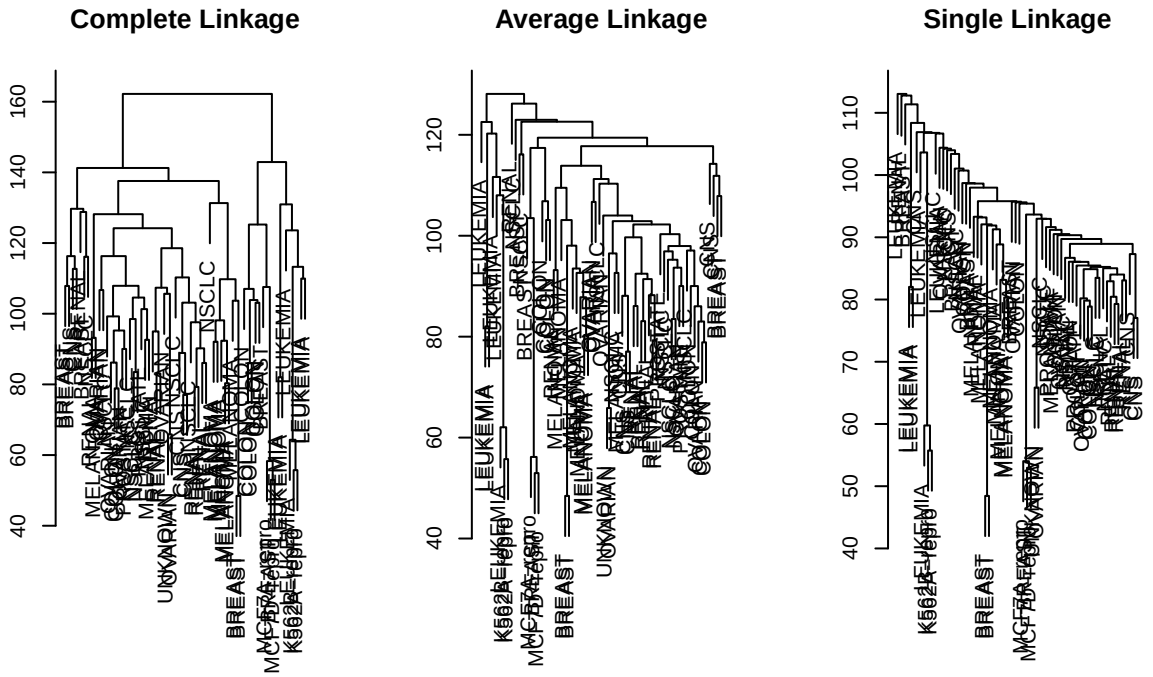
```
pr.out=prcomp(nci.data, scale=TRUE)


# Clustering the Observations of the NCI60 Data

sd.data=scale(nci.data)
par(mfrow=c(1,3))
```

'archical Clustering with Scaled Fte Linkage with Correlation−Base



dist(xsc)
hclust (*, "complete")

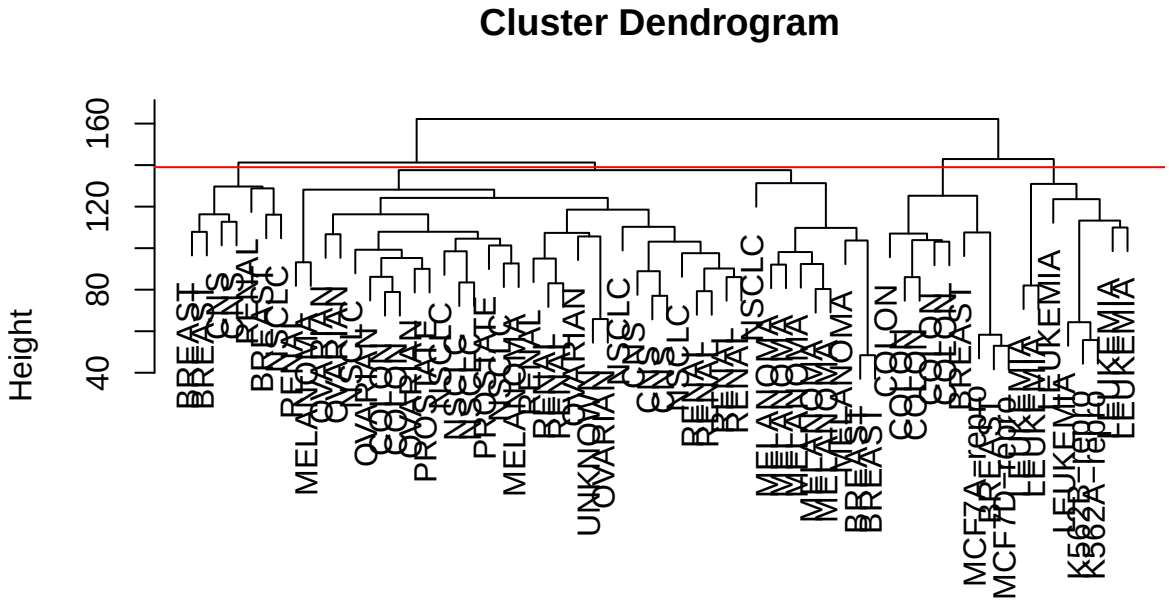
```
data.dist=dist(sd.data)
plot(hclust(data.dist), labels=nci.labs, main="Complete Linkage", xlab="", sub="",ylab="")
plot(hclust(data.dist, method="average"), labels=nci.labs, main="Average Linkage", xlab="", sub="",ylab
plot(hclust(data.dist, method="single"), labels=nci.labs,  main="Single Linkage", xlab="", sub="",ylab=
```




```
hc.out=hclust(dist(sd.data))
hc.clusters=cutree(hc.out,4)
table(hc.clusters,nci.labs)
```

```
##             nci.labs
## hc.clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
##            1      2   3     2           0           0        0           0
##            2      3   2     0           0           0        0           0
##            3      0   0     0           1           1        6           0
##            4      2   0     5           0           0        0           1
##             nci.labs
## hc.clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##            1           0        8     8       6        2     8       1
##            2           0        0     1       0        0     1       0
##            3           0        0     0       0        0     0       0
##            4           1        0     0       0        0     0       0
```

```
par(mfrow=c(1,1))
plot(hc.out, labels=nci.labs)
abline(h=139, col="red")
```



dist(sd.data)
hclust (*, "complete")

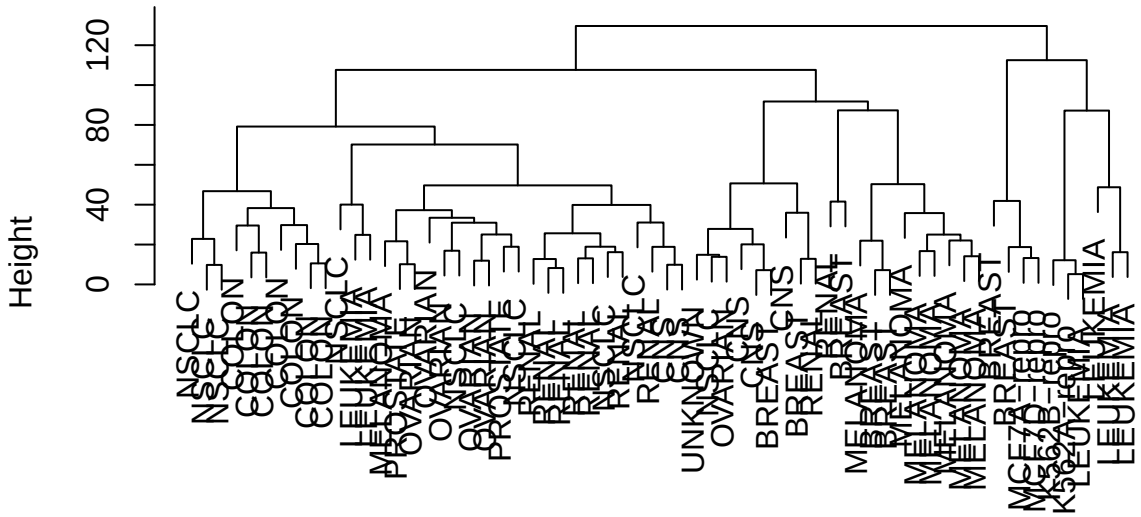
```
hc.out
```

```
##
## Call:
## hclust(d = dist(sd.data))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 64
```

```
set.seed(2)
km.out=kmeans(sd.data, 4, nstart=20)
km.clusters=km.out$cluster
table(km.clusters,hc.clusters)
```

```
##            hc.clusters
## km.clusters  1  2  3  4
##           1 11  0  0  9
##           2 20  7  0  0
##           3  9  0  0  0
##           4  0  0  8  0
```

```
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)
```

```
##    nci.labs
##     BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro
##   1      0   2     7           0           0        2           0
##   2      5   3     0           0           0        0           0
##   3      0   0     0           1           1        4           0
##   4      2   0     0           0           0        0           1
##    nci.labs
##     MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN
##   1           0        1     8       5        2     7       0
##   2           0        7     1       1        0     2       1
##   3           0        0     0       0        0     0       0
##   4           1        0     0       0        0     0       0
```