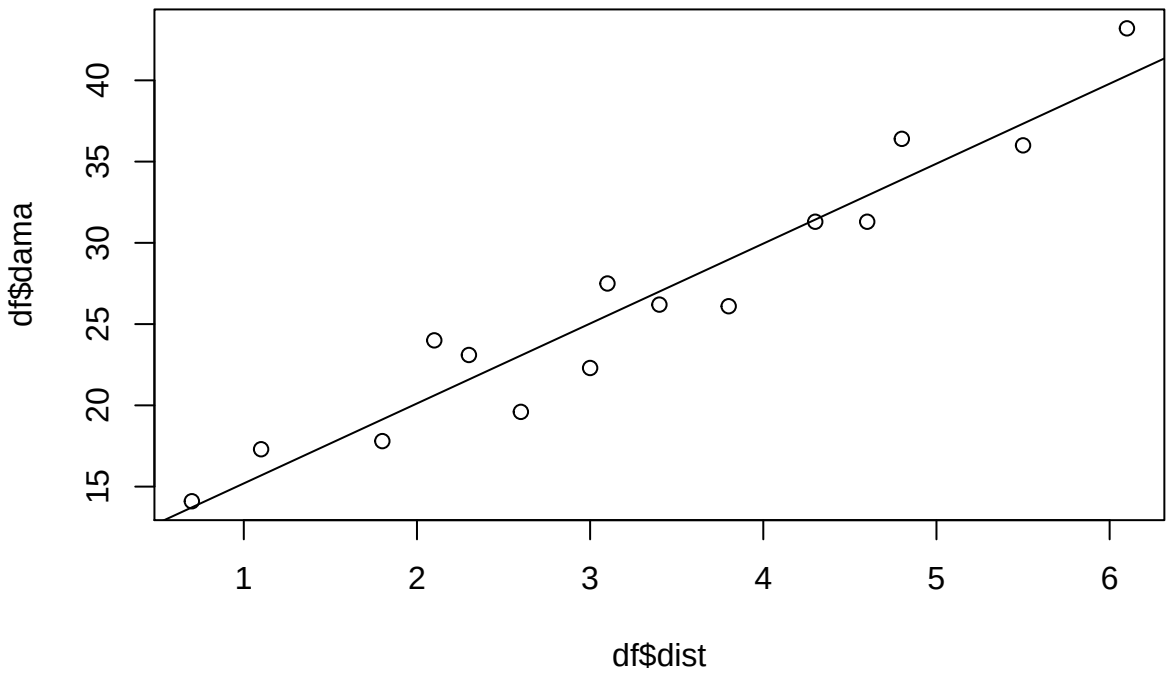
Machine Learning Cheatsheet

Regression

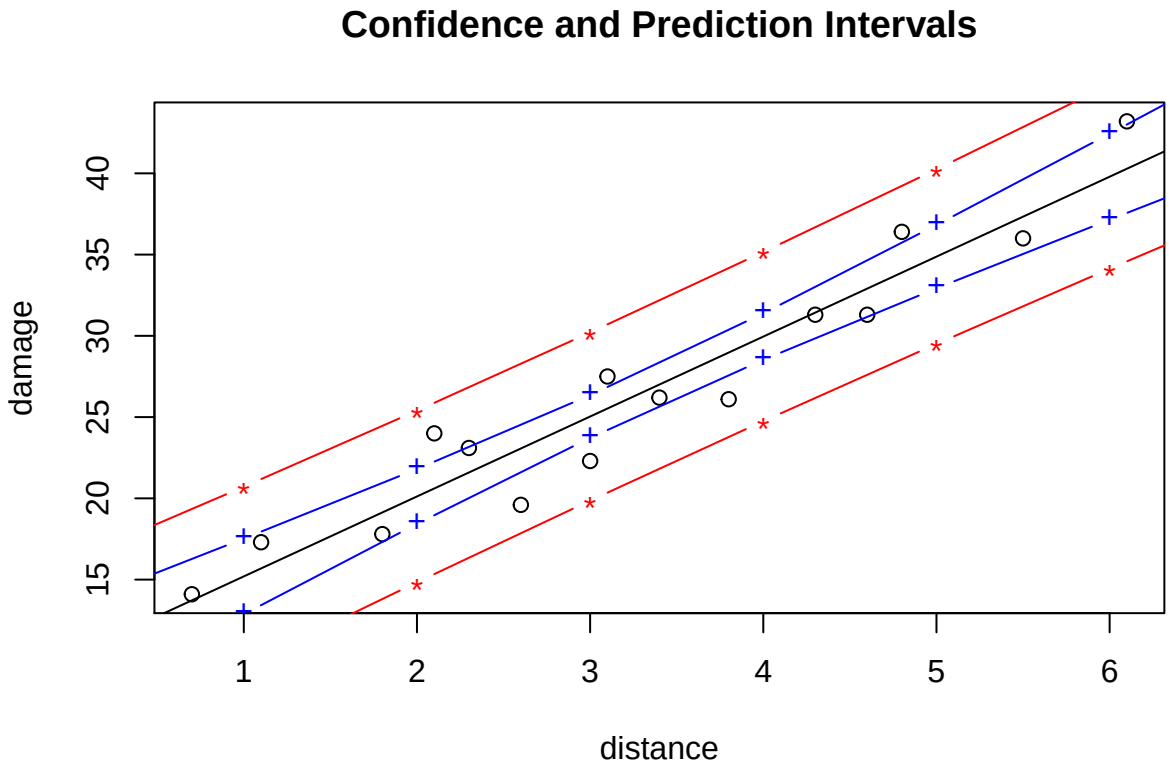
```
dist <- c(3.4,1.8,4.6,2.3,3.1,5.5,0.7,3.0,2.6,4.3,2.1,1.1,6.1,4.8,3.8)
dama <- c(26.2,17.8,31.3,23.1,27.5,36.0,14.1,22.3,19.6,31.3,24.0,17.3,43.2,36.4,26.1)
df <- data.frame(cbind(dist,dama))

m <- lm(dama ~ dist, data = df)
summary(m)
```

```
##
## Call:
## lm(formula = dama ~ dist, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4682 -1.4705 -0.1311  1.7915  3.3915
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.2779     1.4203   7.237 6.59e-06 ***
## dist          4.9193     0.3927  12.525 1.25e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.316 on 13 degrees of freedom
## Multiple R-squared:  0.9235, Adjusted R-squared:  0.9176
## F-statistic: 156.9 on 1 and 13 DF,  p-value: 1.248e-08
```

```
plot(df$dist,df$dama)
abline(m)
```




```
conf <- predict(m, data.frame(dist=seq(0,100)), interval = 'confidence')
pred <- predict(m, data.frame(dist=seq(0,100)), interval = 'prediction')

plot(dist, dama, xlab = 'distance', ylab = 'damage', main = 'Confidence and Prediction Intervals')
abline(m)
lines(seq(0,100),conf[,'lwr'],col = 'blue', type = 'b', pch = '+')
lines(seq(0,100),conf[,'upr'],col = 'blue', type = 'b', pch = '+')
lines(seq(0,100),pred[,'upr'],col = 'red', type = 'b', pch = '*')
lines(seq(0,100),pred[,'lwr'],col = 'red', type = 'b', pch = '*')
```



Logistic Regression

```
mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
summary(mydata)
```

```
##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
mydata$admit <- factor(mydata$admit)
mydata$rank <- factor(mydata$rank)
summary(mydata)
```

```
##  admit       gre             gpa        rank
##  0:273   Min.   :220.0   Min.   :2.260   1: 61
##  1:127   1st Qu.:520.0   1st Qu.:3.130   2:151
##          Median :580.0   Median :3.395   3:121
##          Mean   :587.7   Mean   :3.390   4: 67
##          3rd Qu.:660.0   3rd Qu.:3.670
##          Max.   :800.0   Max.   :4.000
```

```
m <- glm(admit ~ gre + gpa + rank, data = mydata, family = 'binomial')
summary(m)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
##
## Number of Fisher Scoring iterations: 4
```

```
probabilities <- predict(m, mydata, type = 'response')
predictions <- ifelse(probabilities > 0.5, 1, 0)
actuals <- mydata$admit

mean(predictions != actuals)
```

```
## [1] 0.29
```

```
table(predictions, actuals)
```

```
##            actuals
## predictions   0   1
##           0 254  97
##           1  19  30
```

Cross Validation

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.4
```

```
ds <- ISLR::Auto
m <- lm(mpg ~ horsepower, data = ds)
predictions <- predict(m, ds)
mean((ds$mpg-predictions)^2)
```

```
## [1] 23.94366
```

```r
ds <- ISLR::Default
m <- glm(default~balance, data = ds, family = 'binomial')
probabilities <- predict(m, ds, type = 'response')
predictions <- ifelse(probabilities>0.5,1,0)
actuals <- ifelse(ds$default=='Yes',1,0)
table(predictions, actuals)
```

```
##            actuals
## predictions    0    1
##           0 9625  233
##           1   42  100
```

```r
mean(predictions != actuals)
```

```
## [1] 0.0275
```

```r
# Validation Set Approach

ds <- ISLR::Auto
set.seed(1)
split <- sample(nrow(ds),nrow(ds)/2)
train <- ds[split,]
test <- ds[-split,]
m <- lm(mpg~horsepower, data = train)
testpredictions <- predict(m,test)
testmse <- mean((testpredictions-test$mpg)^2); testmse
```

```
## [1] 26.14142
```

```r
# Leave One Out Cross Validation

m <- lm(mpg~horsepower, data = ds)

library(boot)
cverror <- cv.glm(ds, m)
cverror$delta
```

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

```r
# K Fold Cross Validation
cverror <- cv.glm(ds, m, K = 10)
cverror$delta
```

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

Trees

```r
# Regression Trees

ds <- ISLR::Hitters
ds <- na.omit(ds)
library(tree)
```

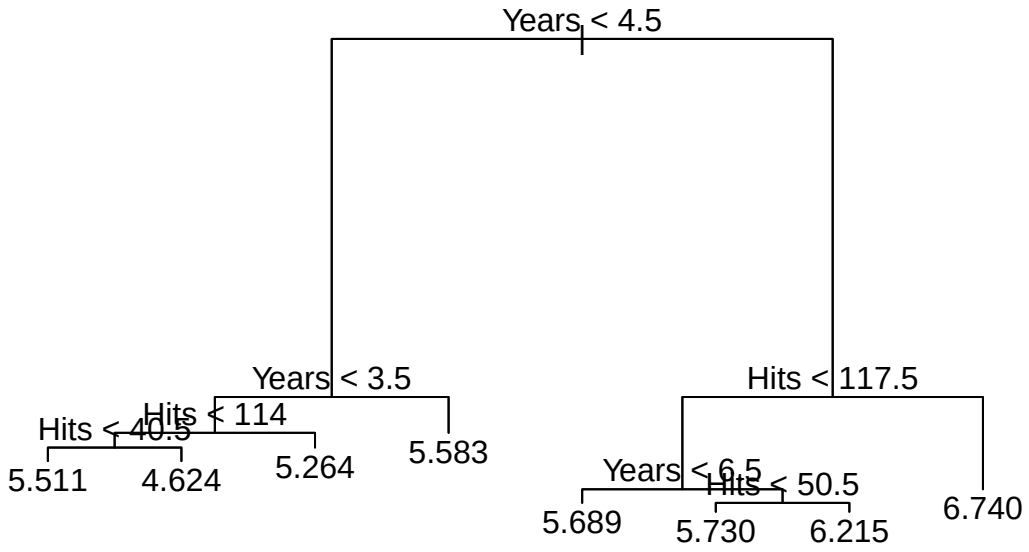
```
## Warning: package 'tree' was built under R version 3.4.4
```

```r
tree.hitters <- tree(log(Salary)~Years+Hits, data = ds)
summary(tree.hitters)
```

```
##
```

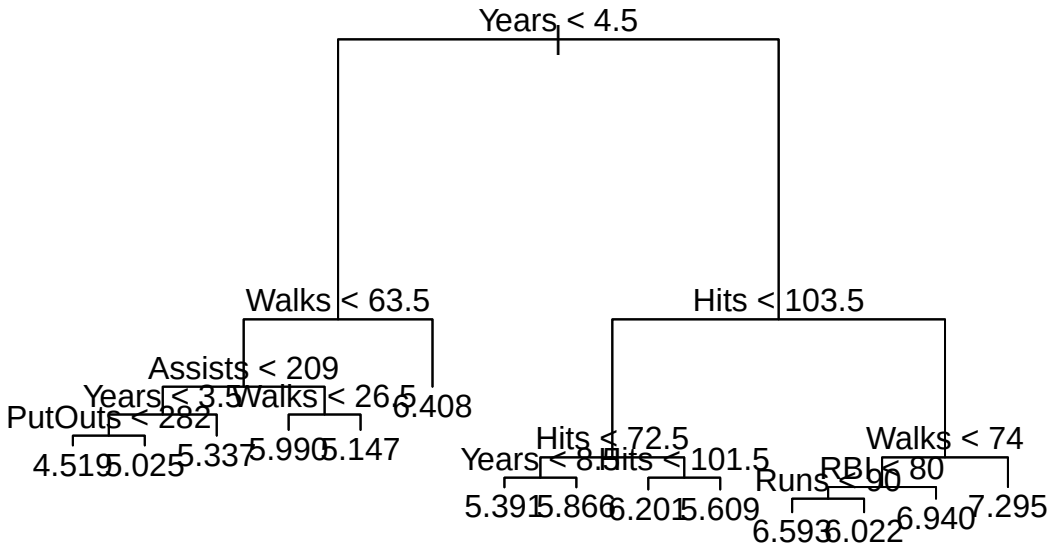
```
## Regression tree:
## tree(formula = log(Salary) ~ Years + Hits, data = ds)
## Number of terminal nodes:  8
## Residual mean deviance:  0.2708 = 69.06 / 255
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.2400 -0.2980 -0.0365  0.0000  0.3233  2.1520
```

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

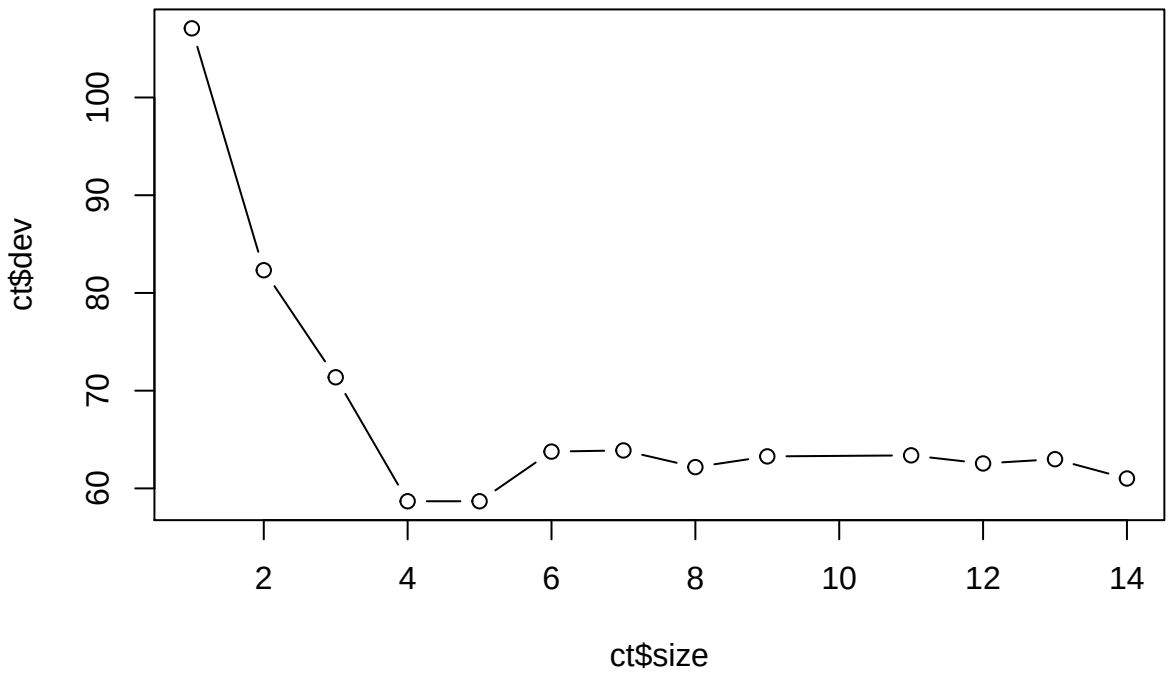



```
y <- predict(tree.hitters, data.frame(Years = 5, Hits = 100))
```

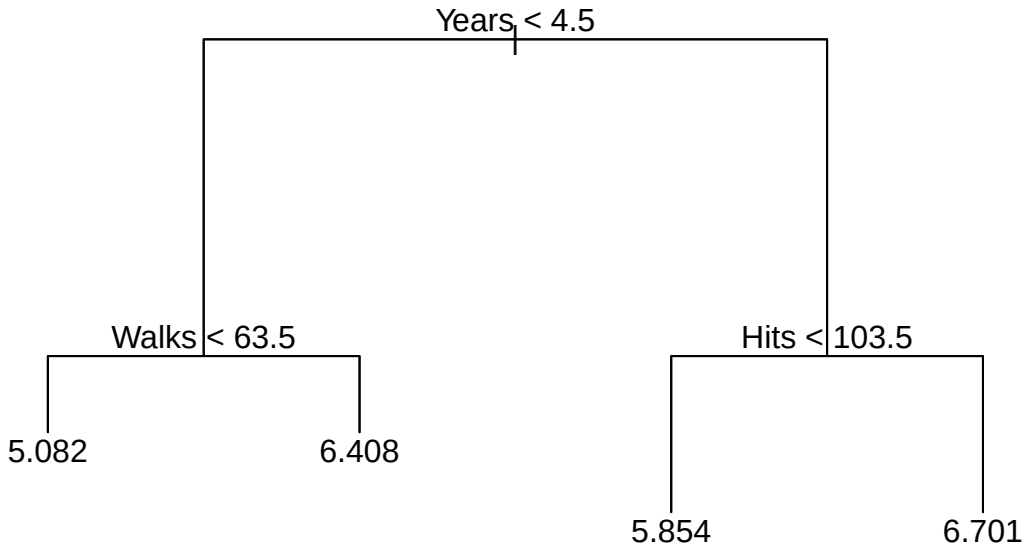
```
# Pruning

set.seed(1)
split <- sample(nrow(ds),132)

train <- ds[split,]
test <- ds[-split,]

t <- tree(log(Salary) ~ Hits + Runs + RBI + Walks + Years + PutOuts + AtBat + Assists + Errors, train)

plot(t)
text(t, pretty = 0)
```




```
ct <- cv.tree(t)
plot(ct$size, ct$dev, type = 'b')
```



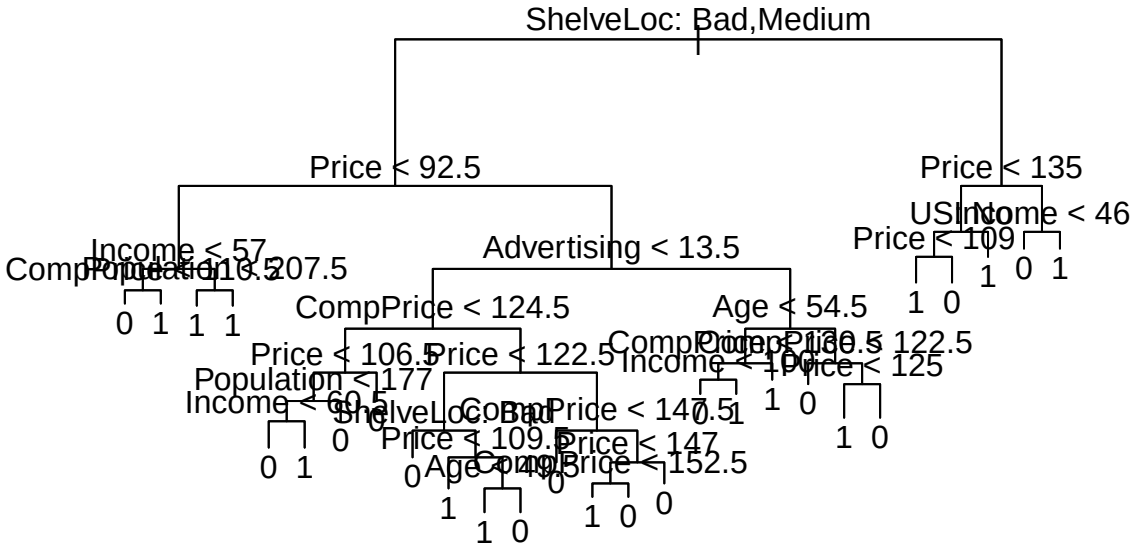

```
pt <- prune.tree(t, best = 4)
plot(pt)
text(pt, pretty = 0)
```




```
# Classification Trees

ds <- ISLR::Carseats

ds$hsales <- factor(ifelse(ds$Sales <= 8,0,1))

t <- tree(hsales ~ . -Sales -hsales, ds)
summary(t)
```

```
##
## Classification tree:
## tree(formula = hsales ~ . - Sales - hsales, data = ds)
## 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(t)
text(t,pretty=0)
```




```
set.seed(2)
indices <- sample(nrow(ds),nrow(ds)/2)

train <- ds[indices,]
test <- ds[-indices,]

t <- tree(hsales ~ . -Sales -hsales, train)
p <- predict(t, test, type = 'class')

set.seed(3)

ct <- cv.tree(t, FUN=prune.misclass) # Cross Validation
pt <- prune.misclass(t, best = 9) # Prune to 9 terminal nodes
```

Bagging & Random Forests

```
library(MASS)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

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

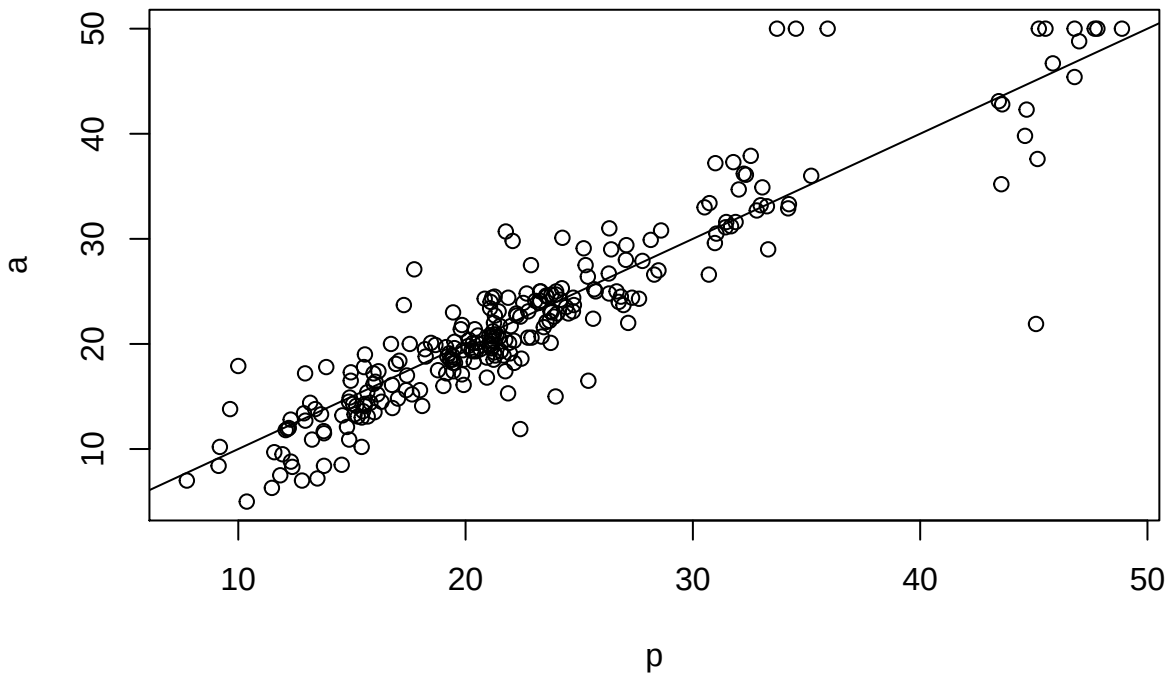
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
d <- MASS::Boston

set.seed(1)
```

```
index <- sample(nrow(Boston),nrow(Boston)/2)
train <- d[index,]
test <- d[-index,]

set.seed(50)
bb <- randomForest(medv ~ .-medv, train, mtry = 13, importance = TRUE); bb
```

```
##
## Call:
##  randomForest(formula = medv ~ . - medv, data = train, mtry = 13,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.14724
##                     % Var explained: 86.5
```

```
p <- predict(bb, test) # Predictions
a <- test$medv # Actuals

plot(p,a)
abline(0,1)
```




```
mse <- mean((p-a)^2); mse
```

```
## [1] 13.30486
```

```r
d <- ISLR::Carseats

d$High <- factor(ifelse(d$Sales<=8,0,1))

set.seed(1)
index <- sample(nrow(d),nrow(d)/2)
train <- d[index,]
test <- d[-index,]

set.seed(2)
bb <- randomForest(High ~ . -Sales - High, train, mtry = 10)

p <- predict(bb, test) # predictions
a <- test$High # actuals

table(p,a)
```

```
##    a
## p     0   1
##   0 104  22
##   1  12  62
```

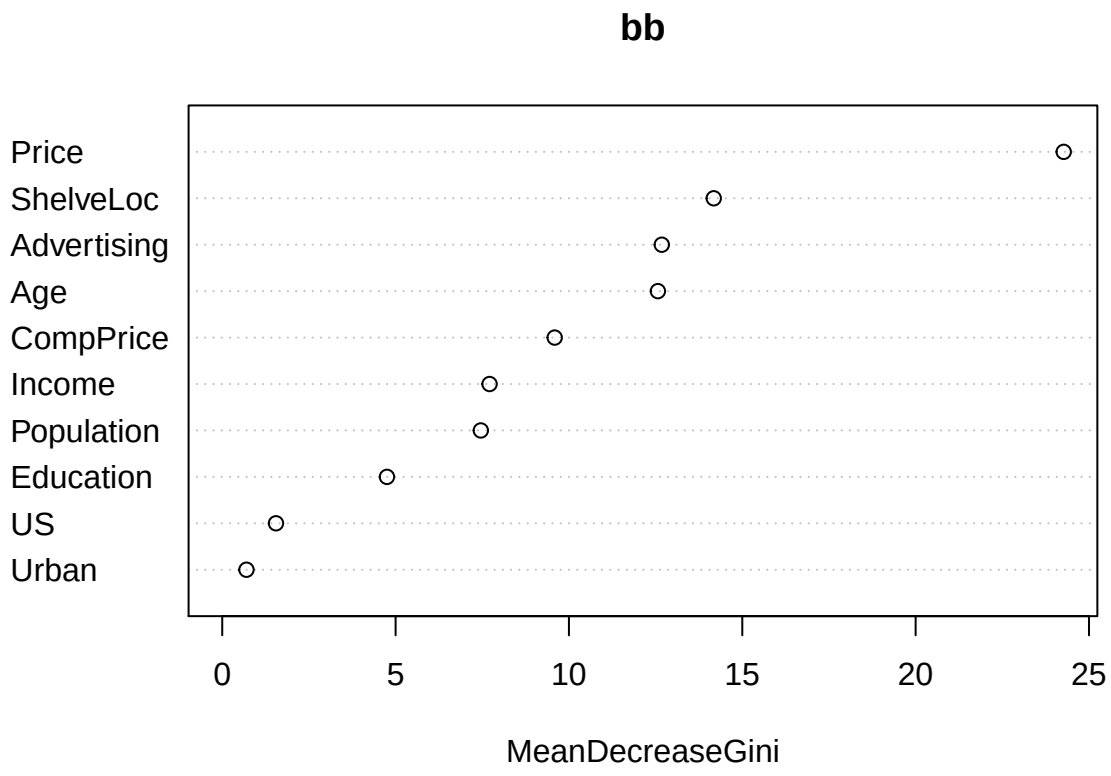
```r
mean(p!=a)
```

```
## [1] 0.17
```

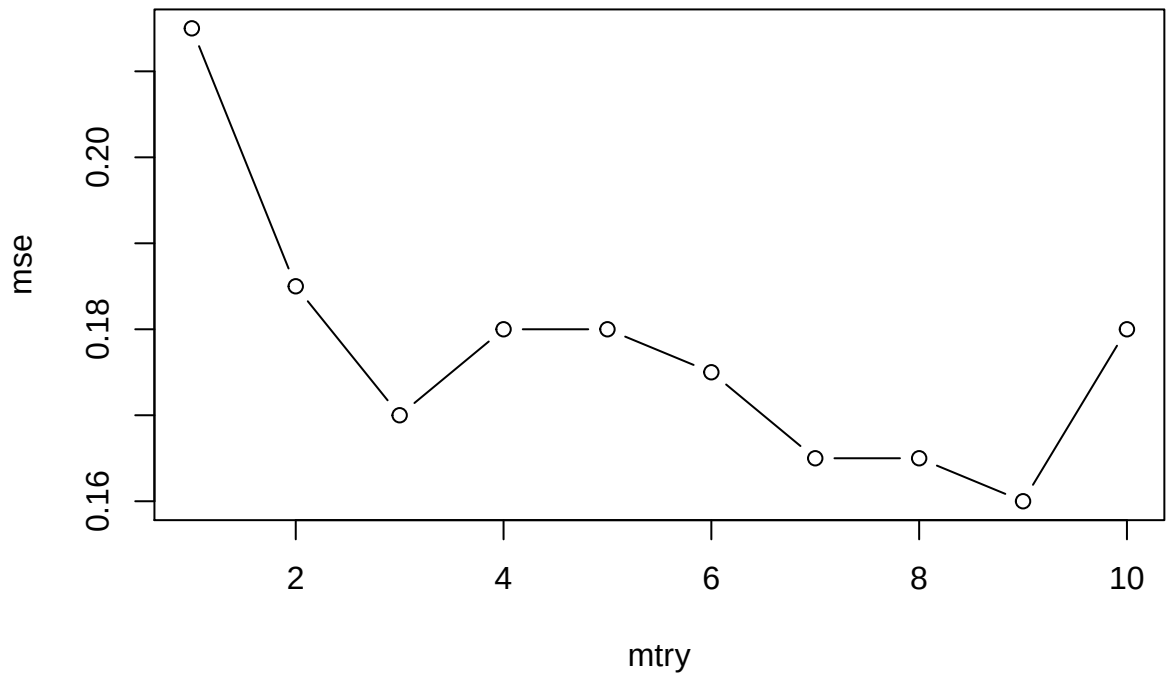
```r
importance(bb)
```

```
##             MeanDecreaseGini
## CompPrice          9.5889890
## Income             7.7103757
## Advertising       12.6777514
## Population         7.4581696
## Price             24.2709465
## ShelveLoc         14.1766331
## Age               12.5653785
## Education          4.7511615
## Urban              0.6988388
## US                 1.5498357
```

```r
varImpPlot(bb)
```




```
# Comparing different levels of mtry

a <- test$High
mse <- rep(0,10)
for (i in 1:10){
  set.seed(5)
  rb <- randomForest(High ~ . -Sales - High, train, mtry=i)
  p <- predict(rb, test)
  mse[i] <- mean((p!=a))
}

plot(mse,xlab='mtry',ylab='mse',type='b')
```



SVMs

```
set.seed(1)
x <- matrix(rnorm(20*2),ncol=2)
y <- c(rep(-1,10),rep(1,10))
d <- data.frame(x=x,y=as.factor(y))

library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```
s <- svm(y ~ ., d, kernel = 'linear', cost = 10, scale = FALSE)
summary(s)
```

```
##
## Call:
## svm(formula = y ~ ., data = d, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  20
```

```
##
##  ( 10 10 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
set.seed(1)
t <- tune(svm, y ~ ., data = d, kernel = 'linear', ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100)))
summary
```

```
## function (object, ...)
## UseMethod("summary")
## <bytecode: 0x000000001548de70>
## <environment: namespace:base>
```

```
bm <- t$best.model
summary(bm)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = d, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  20
##
##  ( 10 10 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
s <- svm(y ~ ., d, kernel = 'radial', gamma = 1, cost = 10, scale = FALSE)
summary(s)
```

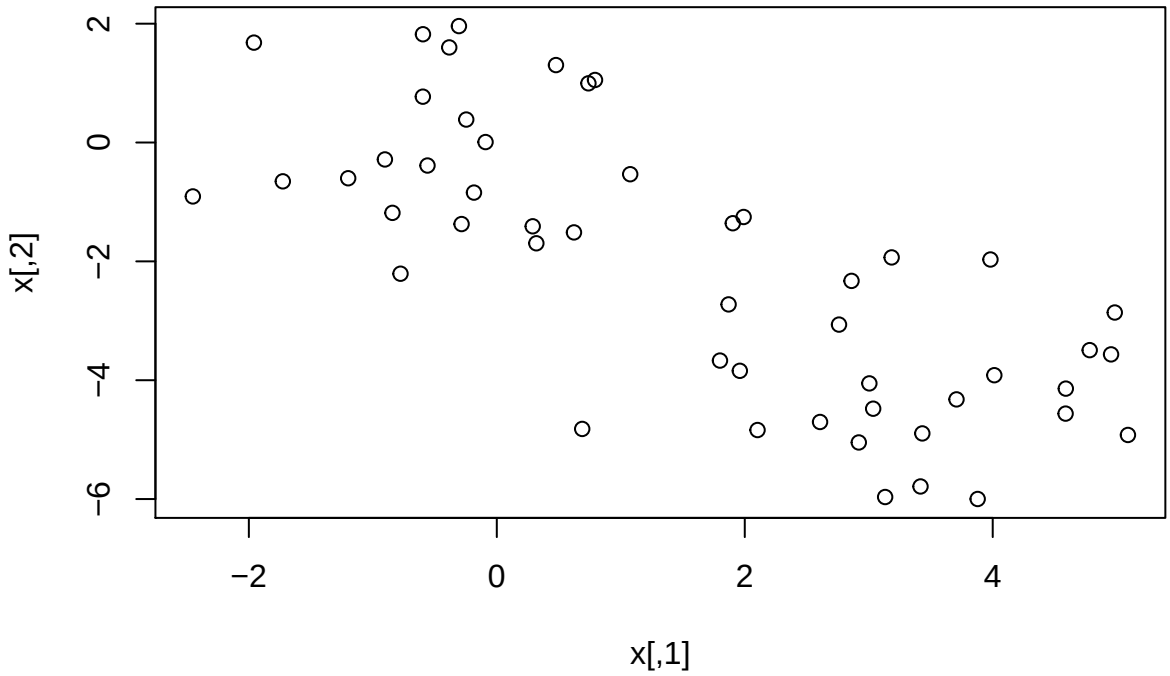
```
##
## Call:
## svm(formula = y ~ ., data = d, kernel = "radial", gamma = 1,
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
```

```
##       gamma:  1
##
## Number of Support Vectors:  17
##
##  ( 9 8 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
set.seed(1)
t <- tune(svm, y ~ ., data = d, kernel = 'radial', ranges = list(cost = c(0.1,1,1,10,100,1000), gamma =
```

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
plot(x)
```




```
k <- kmeans(x,2,nstart=20); k
```

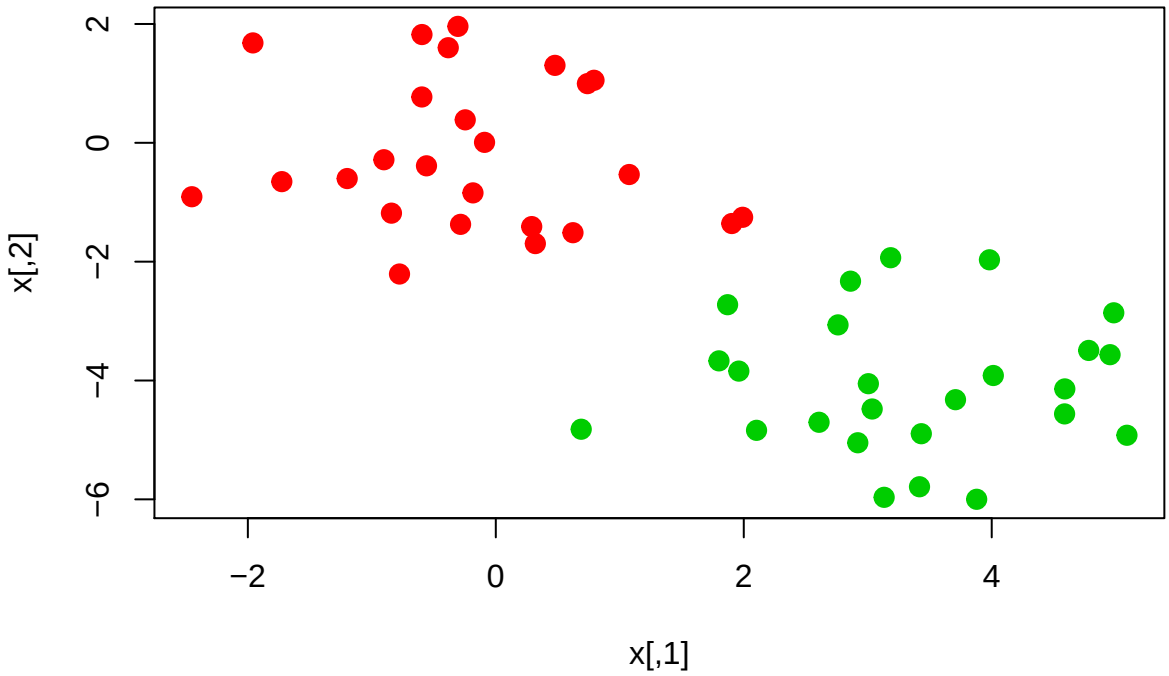
```
## K-means clustering with 2 clusters of sizes 25, 25
##
```

```
## Cluster means:
##         [,1]       [,2]
## 1 -0.1956978 -0.1848774
## 2  3.3339737 -4.0761910
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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
##
## Within cluster sum of squares by cluster:
## [1] 65.40068 63.20595
##  (between_SS / total_SS =  72.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
k$cluster # Assigned Clusters
```

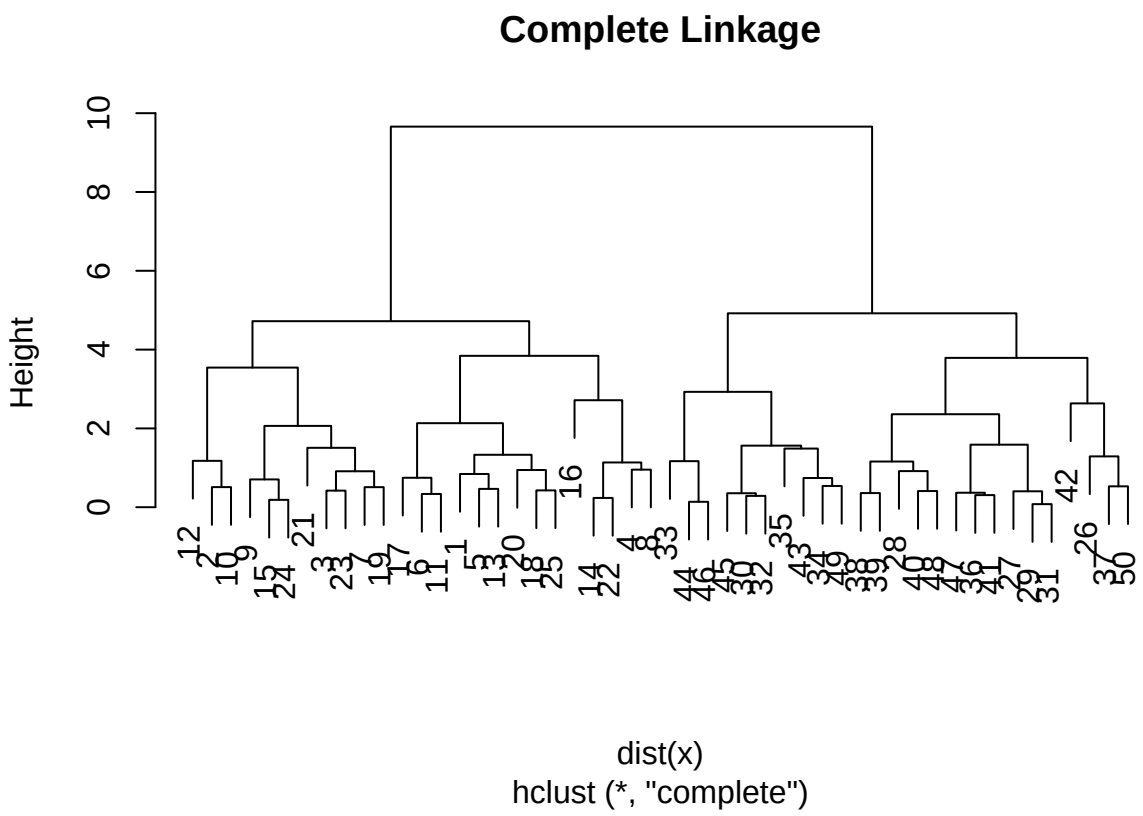
```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
plot(x, col=(k$cluster+1), pch = 20, cex = 2)
```




```
hc <- hclust(dist(x),method = 'complete')
ha <- hclust(dist(x),method = 'average')
```

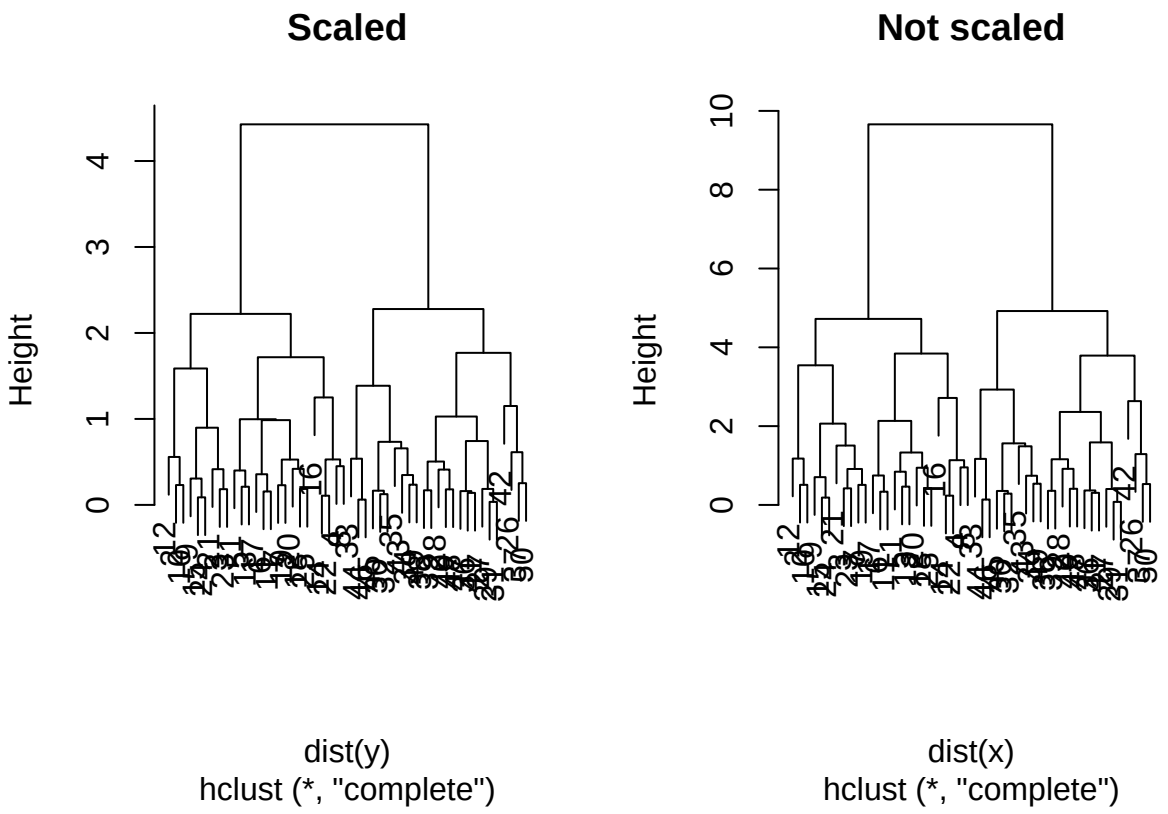
```
hs <- hclust(dist(x),method = 'single')

plot(hc, main = 'Complete Linkage')
```




```
cutree(hc,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 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
```

```
y <- scale(x)
par(mfrow=c(1,2))
plot(hclust(dist(y),method='complete'), main = 'Scaled')
plot(hclust(dist(x),method='complete'), main = 'Not scaled')
```



Principal Component Analysis

```
x <- c(2.5,0.5,2.2,1.9,3.1,2.3,2,1,1.5,1.1)
y <- c(2.4,0.7,2.9,2.2,3.0,2.7,1.6,1.1,1.6,0.9)
d <- data.frame(x,y)

p <- prcomp(d, scale = FALSE); p
```

```
## Standard deviations (1, .., p=2):
## [1] 1.1331495 0.2215477
##
## Rotation (n x k) = (2 x 2):
##          PC1        PC2
## x -0.6778734  0.7351787
## y -0.7351787 -0.6778734
```

```
p$rotation
```

```
##          PC1        PC2
## x -0.6778734  0.7351787
## y -0.7351787 -0.6778734
```

```
p$center # centre of the new coordinates
```

```
##    x    y
## 1.81 1.91
```

```
p$scale
```

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

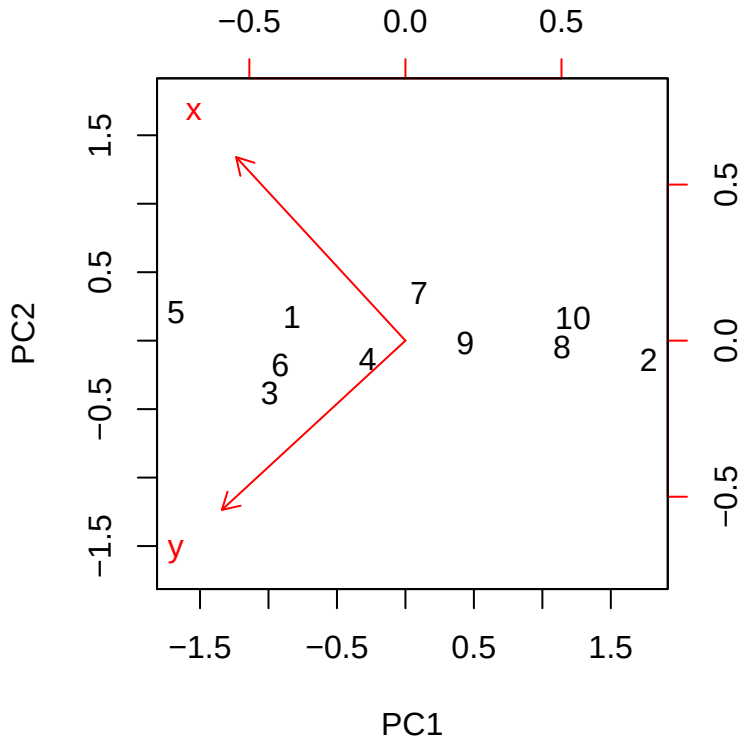
```
p$sdev
```

```
## [1] 1.1331495 0.2215477
```

```
p$sdev^2 # eigenvalues
```

```
## [1] 1.2840277 0.0490834
```

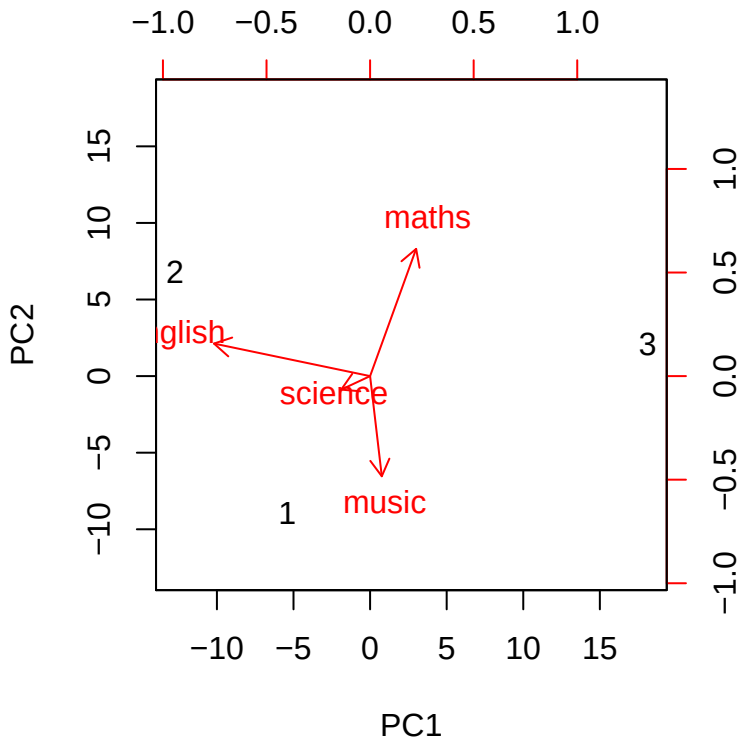
```
biplot(p, scale = FALSE)
```



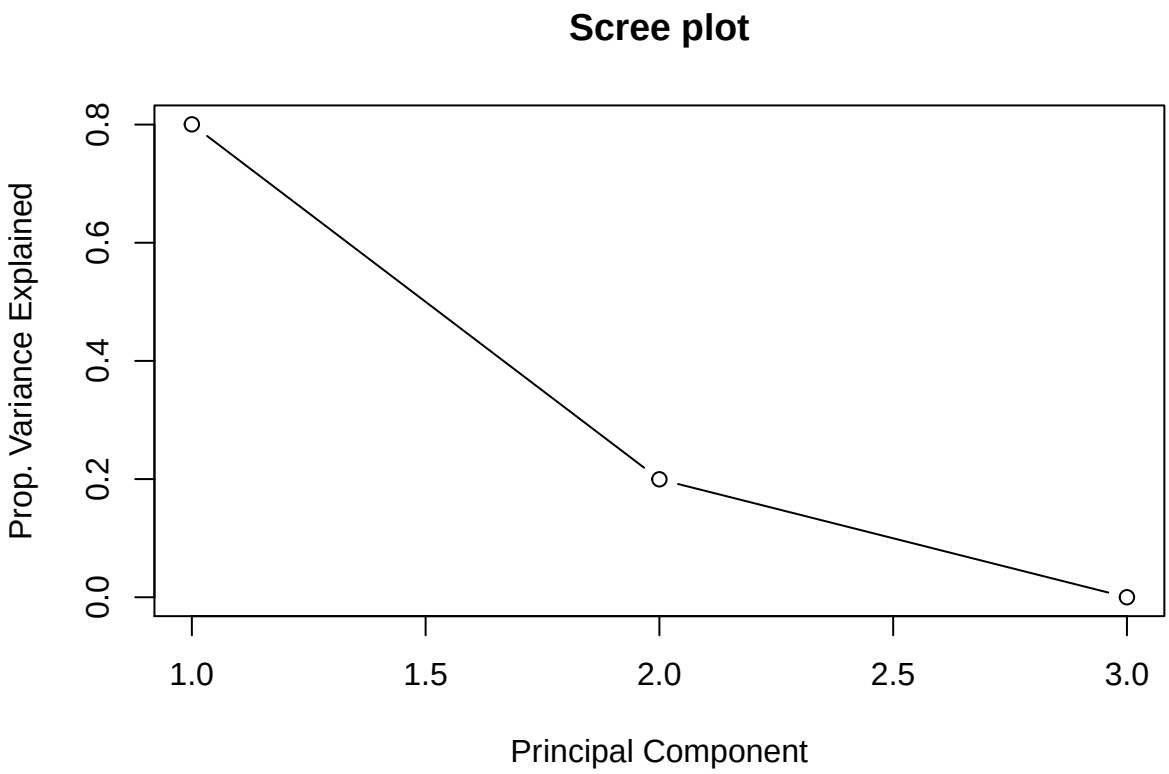

```
d <- data.frame(maths = c(80,90,95),science = c(85,85,80),
                english = c(60,70,40), music = c(55,45,50))
p <- prcomp(d, scale = FALSE)
p$rotation
```

```
##                 PC1         PC2       PC3
## maths    0.27795606  0.76772853 0.5616653
## science -0.17428077 -0.08162874 0.4099911
## english -0.94200929  0.19632732 0.1426679
## music    0.07060547 -0.60447104 0.7043332
```

```
biplot(p, scale = FALSE)
```




```
pve <- p$sdev^2/sum(p$sdev^2)
plot(pve, xlab = 'Principal Component', ylab = 'Prop. Variance Explained', main = 'Scree plot', type =
```


Scree plot
Prop. Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
1.5
2.0
2.5
3.0
Principal Component