Chapter 2: Statistical Learning

Chapter 2: What is statistical learning

We observe a quantitative response Y and p different predictors, $X_1, X_2, ... X_p$. We assume a relationship between Y and X which can be written in the general form:

$$Y = f(X) + \epsilon \tag{2.1}$$

f represents the unknown function of X and δ represents the error term. F is the systematic information that X provides about Y.

Why do we estimate y?

1. **Prediction** *$\hat{f}$ is the estimate for f and $\hat{Y}$ is the prediction the prediction for Y.

- *reducible error*: inaccuracy with predictions
- *irreducible error*: Y is a function of δ and therefore cannot reduce the error introduced by δ

$$\hat{Y} = \hat{f}(X) \tag{2.2}$$

$$E(Y - \hat{Y})^2 = E[f(X) + \epsilon - \hat{f}(X)]^2 = [f(X) - \hat{f}(X)]^2 + Var(\epsilon)$$

The first term $[f(X) - \hat{f}(X)]^2$ is the reducible error and the variance of epsilon is the irreducible error.

2. **Inference** *This is where we want to estimate but not predict f.

- 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?

How do we estimate f?

Training data is a set of observations used to teach a method to estimate f.

We can use two models to estimate f.

Parametric models There models have a two step approach: - First Assume f:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{2.4}$$

This is a linear model where instead of having to estimate an entirely arbitrary p-dimensional function $f(X)$ we only need to estimate $p + 1$ coefficients. * fit/train the model

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

We need to estimate the parameter such that the approximate Y. This is called *ordinary least squares*.

This procedure is called parametric because estimating f is through the estimation of a set of parameters. While much simpler to do, this can cause approximation variance.

Flexible models can fit many functional forms for f. *Over fitting* can occur if the model is too complex and follows *noise* too closely.

Nonparametric models Nonparametric models don't make explicit assumptions but seek estimates of f. While we don't need to worry about fit of data, a large amount of data is needed to obtain an accurate estimate of f.

Thin-plate spline is used to estimate f by not imposing a predefined model on f but attempts to produce estimates for f that is as close to observed data.

This observed fit is called *smoothness*. The smoother the visualization fo the data is, the better fit. Choosing the correct metric of smoothing to avoid over-fitting will be discussed.

Extra: Smoothness measure

Data fidelity term The TPS smoothness measure arises from considering the integral of the second derivative (usually denoting the curvature/concavity of a graph). In this case, we can use the *energy function* (a function we want to minimize/maximize):

$$E_{tps}(f) = \sum_{i=1}^{K} ||y_i - f(x_i)||^2$$

Known as the *data fidelity term*, this measures how well function f fits the data points (x_i, y_i). The goal is to minimize the difference or error of the observed and true values of y_i and$f(x_i)$.

Smoothness Penalty Term The smoothness variant uses a tuning parameter λ (which is externally constructed) to control the "rigidity" and minimize $E_{tps,smooth}$ with a unique minimizer f:

$$E_{tps,smooth}(f) = \sum_{i=1}^{K} ||y_i - f(x_i)||^2 + \lambda \int\int [(\frac{\partial^2}{\partial x_1^2})^2 + 2(\frac{\partial f}{\partial x_1 x_2})^2 + (\frac{\partial^2 f}{\partial x_2^2})^2] dx_1 dx_2$$

This penalty term includes the sum of squared second order partial derivatives that quantify the curvature of f in different directions such as: * $\frac{\partial^2}{\partial x_1^2}$: curvature in x_1 direction * $\frac{\partial f}{\partial x_1 x_2}$ mixed curvature between x_1 and x_2 * $\frac{\partial^2 f}{\partial x_2^2}$ curvature in x_2 direction

The trade-off between prediciton accuracy and model interpretability

Thin plate splines can generate much wider ranges of possible shapes to estimate f.

A more restrictive model is used for inference whereas linear models are good for prediction. Splines can lead to super complicated estimates of f that lead to difficult predictor separation to analyze the effect on the response.

Other models Restrictive - linear regression - thin plate splines - lasso - GAM Flexible - thin plate splines - bootstrapping - bagging, boosting, support vector machines

Supervised vs unsupervised learning

Supervised learning is used when we wish to fit a model that relates the response to the predictors with the aim of accurately predicting the response for future observations. Examples include - linear/logistic regression - GAM - boosting, support vector machines

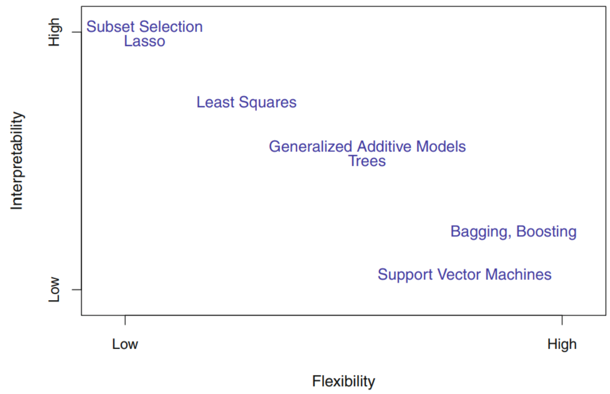


Figure 1: Representation of tradeoff

Unsupervised learning is used when we observe a vector of measurements x_i but no associated response y_i. We lack a response to supervise our analysis so instead we understand relationships between variables and observations. Examples - cluster analysis

Semi supervised learning is used when we have disjoint sets of supervised an unsupervised observations. We wish to incorporate m observations which response measurements are available and $n - m$ observations for which they are not.

Regression vs. classification problems

Variables can be characterized as either *quantitative* or *qualitative/categorical*.

We tent o prefer regression for quantitative (ex: least squares regression) and classification for categorical variables (ex: logistic regression). Some methods like K-nearest neighbors and boosting used for both cases.

2.2 Assessing model accuracy

Measuring the quality of fit

We need to quantify the extent to which the predicted response vlaue for a given observation is close t the true response value for that observation.

This is called the *Mean Squared Error*

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{2.5}$$

$\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the ith observation. The MSE is computed from the training data.

training MSE is used when we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen test data.

Described mathematically: suppose we use our training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ to estimate $\hat{f}$. We can then compute $\hat{f}(x_1), \ldots, \hat{f}(x_n)$. If our response is close to the true response, then our MSE is small. But we don't want to know if $\hat{f}(x_i) \approx y_i$.

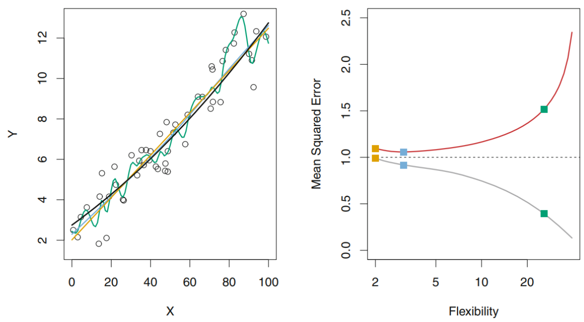
We want to know whether $\hat{f}_0$ is approx equal to y_0 where (x_0, y_0) is previously unseen data. We need to choose the method that gives the lowest *test MSE* as opposed to the lowest training MSE.

We can use the *average squared prediction error*

$$Ave(y_0 - \hat{f}(x_0))^2 \tag{2.6}$$

We want the method with the smallest average test MSE.

What if we don't have test observations to find the smallest test MSE? We can't use the training MSE because there is no guarantee that the method with the lowest training MSE will also have the lowest test MSE. The problem is that many methods estimate coefficients to minimize the training set and ignore testing measures.



The true f is approx linear. We observe the training MSE decreases monotonically as the model flexibility increases. But because the truth is closer to linear, the test MSE only slightly decreases before increasing again. Therefore linear regression is better than splicing.

The flexibility level corresponding to the model with the minimal test MSE can vary considerably among data sets. Using *cross validaiton* can be sued to estimate the test MSE using the training data.

Bias-Varinace trade off

The U-shapes MSE's is the consequence of two competing properties of statistical learning methods. It can be shown that the expected test MSE for a given value x_0 can always eb decomposed intot eh sum of three fundamental quantities: - variance of $\hat{f}(x_0)$ - squared bias of $\hat{f}(x_0)$ - variance of error term ϵ

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon) \tag{2.7}$$

$E(y_0 - \hat{f}(x_0))^2$ is the *expected test MSE* and refers to the average test MSE that we would obtain if we repeatedly estimated f using large number of training sets. - the overall expected MSE can be comuted by averaging the exptected test MSE over all possible values of x_0 in the test set

2.7 shows that to minimize the expected test MSE we need to minimize variance and bias. $Var(\epsilon)$ is non-zero and irreducible, therefore the expected test MSE can never be smaller than the variance of ϵ.

Variance is the amount in which $\hat{f}$ would change if using a different training data set. *Bias* is the error that is introduced by approx real life problem. Generally more flexible methods result in less bias as the variance increases while the bias decreases. The rate of change between the two determines the increase or decreae of the test MSE.

The classification setting

The common approach of quantifying the accuracy of our estimate with categorical data is the training *error ate*, which is the proportion of mistakes that are made if we apply our estimate $\hat{f}$ to the training observations:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{2.8}$$

$I(y_i \neq \hat{y_i})$ is the indicator variable that equals 1 if $y_i \neq \hat{y_i}$ and 0 otherwise. 0 means that the variable was classified correctly and vice versa.

2.8 is the training error because if is computed based on the training data. We are most interested in the average *test* error over the set of *test observations* of the form (x_0, y_0) given be

$$Ave(y_i \neq \hat{y_i}) \tag{2.9}$$

The Bayes Classifier It is possible to show that the test error rate given in 2.9 is miniized on average by a simple classifier that assigns each observation to the most likely case given its predictor values.

In other words: simply assign a test observation with predictor value x_0 to the class j for which

$$Pr(Y = j|X = x_0) \tag{2.10}$$

The proof behind the pudding Citation: Shuzhan Fan **The Bayes theorem** Given a feature vector $X = (x_1, \dots, x_n)$ and a class variable C_k, the Bayes theorem states that:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} \text{ for k = 1, 2, ...K}$$

$P(C_k|X)$ is the posterior probability and $P(X|C_k)$ is the likelihood. $P(C_k)$ is the prior probability of class and $P(X)$ is the probability of the predictor.

Using the chain rule the likelihood $P(X|C_k)$ can be decomposed as:

$$P(X|C_k) = P(x_1 \dots x_n|C_k) = P(x_1|x_2, \dots, x_n, C_k)P(x_2|x_3, \dots, x_n, C_k) \dots P(x_{n-1}|x_n, C_k)P(x_n|C_k)$$

Naive independence assumption

The naive conditional independence assumption allows us to calculate $P(X|C_k)$ easily by:

$$P(x_i|x_{i+1}, \dots, x_n|C_k) = P(x_1|C_k)$$

We can get:

$$P(X|C_k) = P(x_1, \dots, x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

The posterior probability $P(C_k|X)$ can be written now as:

$$P(C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(X)}$$

Naive Bayes model Because the predictor $P(X)$ is constant given the input, we can get $P(C_k|X)$ which is positively proportional to:

$$P(C_k|X) \propto P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

Therefore the naive Bayes classification for different class values of C_k looks at the maximum of $P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$ which can be formulated to:

$$\hat{C} = argmax_{C_k} P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

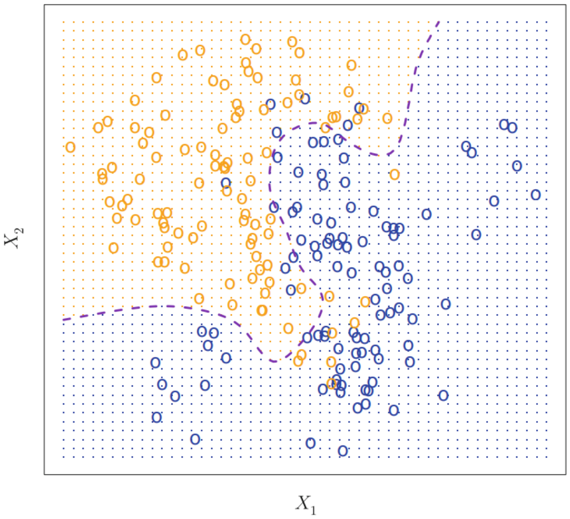
The prior probability of class $P(C_k)$ could be calculated as the relative frequency of class C_k in the training data.

The difference between naive Bayes vs Bayes The likelihood $P(x_i|C_k)$ is usually modeled using the same class distributions, but for the naive Bayes classifier the assumption is made regarding the distribution of $P(x_i|C_k)$.

A conditional probability is represented as the probability of $Y = j$ given the observed predictor vector x_0. This complete classifier is called the *Bayes classifier*.

In a two class problem where there is only class 1 and class 2, the Bayes classifier categorizes classes by a parameter like $P(Y = 1|X = x_0) > 0.5$.

The *Bayes decision boundary* is the line that represents the points where the probability is exactly 50%. The Bayes classifier prediction is determined by the Bayes decision boundary. The image below illustrates this:



The Bayes classifier produces the lowest possible test error rate, called the *Bayes error rate*. Because the Bayes classifier will always choose the class which the conditional probability is the largest, the error rate $X = x_0$ will be $1 - max_j P(Y = j|X = x_0)$.

The overall Bayes error is given by

$$1 - E(max_j P(Y = j|X)) \tag{2.11}$$

Because the classes overlap in the true population in the image above, $max_j P(Y = j|X = x_0) < 1$ for some values of x_0. *The Bayes error rate is analogous to the irreducible error.*

K-nearest neighbors

For real data we don't have conditional distributions of Y given X. so computing the Bayes classifier is impossible. Many approaches attempt to estimate Bayes estimator as the gold standard and then classify a given observation to the class with the highest *estimated* probability.

A method that does this is called the KNN classifier. Given a positive integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0 represented by $\mathcal{N}$.

It estimates the conditional probability for the class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{2.12}$$

Finally the KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability.

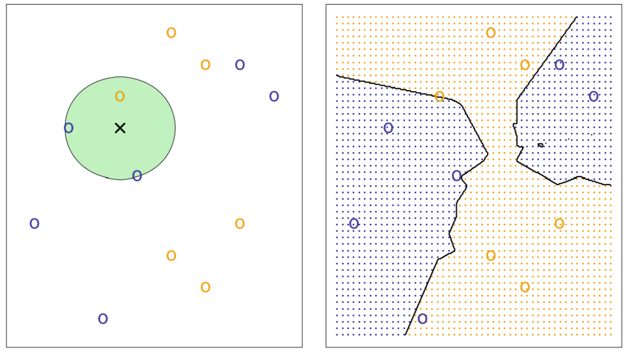


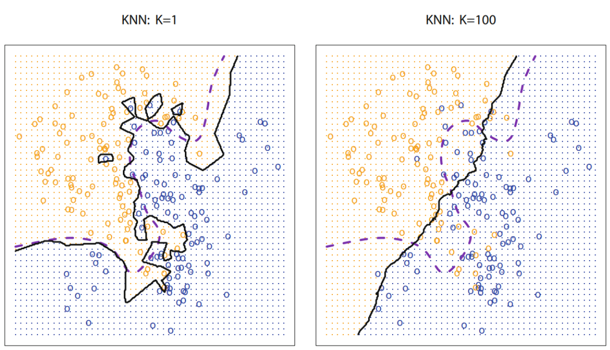
Figure 2: The KNN Classifer

In the example below we set $K = 3$ at all possible values for X_1 and X_2. The KNN decision boundary is illustrated in green. The black cross illustrates the test observation at which a predicted class label is desired. - the three closest points are identified and classified to blue: the most commonly occurring class

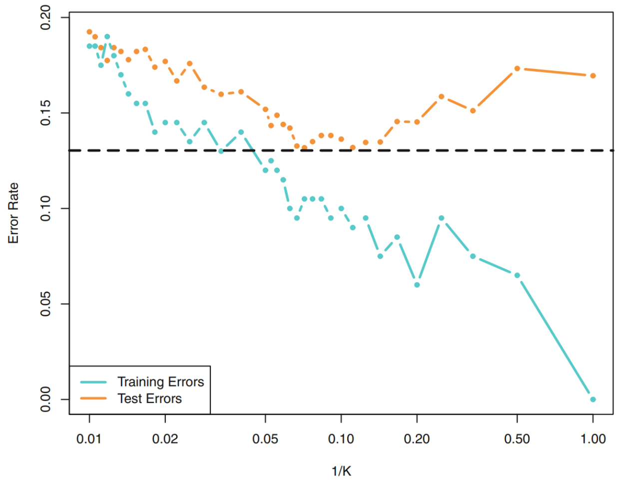
The KNN boundary in black splits blue and orange class observation regions.

The choice of K The choice of K has a drastic effect on the KNN classifier. When $K = 1$ the decision boundary is overly flexible and over fits the data. Logically, this corresponds to a classifier that has low bias but very high variance. As K grows the method becomes less flexible and produces a decision boundary that is close to *linear*. - *this gives us a low variance but high bias classifier*

In general as we use more flexible classification methods the training error rate will decline but the test error rate may not.



With $K = 1$ the decision boundary is too flexible and vice versa for $K = 100$.



The blacked dashed line is the Bayes error rate. The error rates are measured as the level of flexibility, accessed by $\frac{1}{K}$, increases, (ie the number of neighboring Ks decrease). *The more flexible the less neighboring* Ks

Excersises

1.

a) flexible
b) inflexible
c) inflexible
d) flexible

Usually, inflexible methods are used for predicting with large numbers of predictor p or with a strong relationship. Flexible methods are for large sample sizes that correspond to high variance but low bias. In a case with a large n and small p, we can use flexible methods without the fear of overfitting.

2.

a) regression, inference
b) classification, inference
c) regression, prediction

Applied

```
# read csv
college <- read.csv("C:/Users/chuan_71/OneDrive/Desktop/TRADING/ISL/College.csv")

# data exploration
head(college)
```

```
##                              X Private Apps Accept Enroll Top10perc Top25perc
## 1 Abilene Christian University     Yes 1660   1232    721        23        52
## 2           Adelphi University     Yes 2186   1924    512        16        29
## 3               Adrian College     Yes 1428   1097    336        22        50
## 4          Agnes Scott College     Yes  417    349    137        60        89
```

```
## 5    Alaska Pacific University     Yes  193    146     55         16         44
## 6            Albertson College     Yes  587    479    158         38         62
##   F.Undergrad P.Undergrad Outstate Room.Board Books Personal PhD Terminal
## 1        2885         537     7440       3300   450     2200  70       78
## 2        2683        1227    12280       6450   750     1500  29       30
## 3        1036          99    11250       3750   400     1165  53       66
## 4         510          63    12960       5450   450      875  92       97
## 5         249         869     7560       4120   800     1500  76       72
## 6         678          41    13500       3335   500      675  67       73
##   S.F.Ratio perc.alumni Expend Grad.Rate
## 1      18.1          12   7041        60
## 2      12.2          16  10527        56
## 3      12.9          30   8735        54
## 4       7.7          37  19016        59
## 5      11.9           2  10922        15
## 6       9.4          11   9727        55
```

```r
college = college[,-1]

summary(college)
```

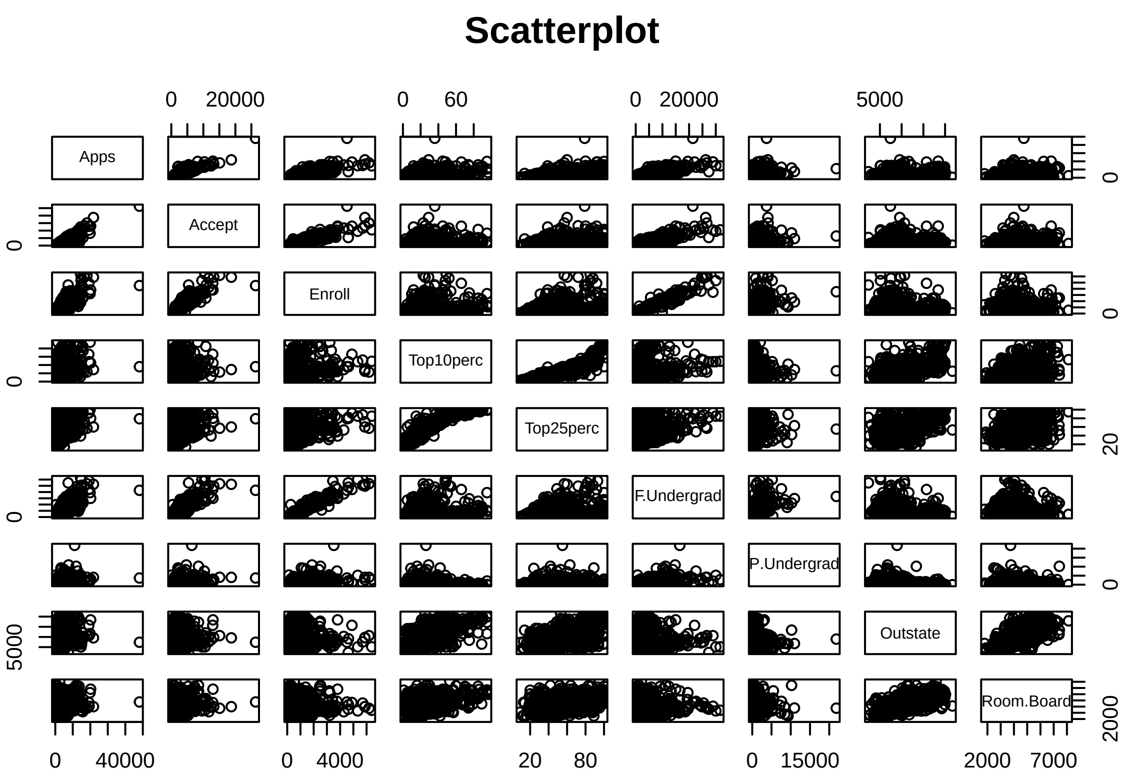
```
##    Private                Apps           Accept          Enroll    
##  Length:777         Min.   :   81   Min.   :   72   Min.   :  35  
##  Class :character   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242  
##  Mode  :character   Median : 1558   Median : 1110   Median : 434  
##                     Mean   : 3002   Mean   : 2019   Mean   : 780  
##                     3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902  
##                     Max.   :48094   Max.   :26330   Max.   :6392  
##    Top10perc       Top25perc      F.Undergrad     P.Undergrad     
##  Min.   : 1.00   Min.   :  9.0   Min.   :  139   Min.   :    1.0  
##  1st Qu.:15.00   1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0  
##  Median :23.00   Median : 54.0   Median : 1707   Median :  353.0  
##  Mean   :27.56   Mean   : 55.8   Mean   : 3700   Mean   :  855.3  
##  3rd Qu.:35.00   3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0  
##  Max.   :96.00   Max.   :100.0   Max.   :31643   Max.   :21836.0  
##     Outstate       Room.Board       Books           Personal   
##  Min.   : 2340   Min.   :1780   Min.   :  96.0   Min.   : 250  
##  1st Qu.: 7320   1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850  
##  Median : 9990   Median :4200   Median : 500.0   Median :1200  
##  Mean   :10441   Mean   :4358   Mean   : 549.4   Mean   :1341  
##  3rd Qu.:12925   3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700  
##  Max.   :21700   Max.   :8124   Max.   :2340.0   Max.   :6800  
##       PhD            Terminal       S.F.Ratio      perc.alumni   
##  Min.   :  8.00   Min.   : 24.0   Min.   : 2.50   Min.   : 0.00  
##  1st Qu.: 62.00   1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00  
##  Median : 75.00   Median : 82.0   Median :13.60   Median :21.00  
##  Mean   : 72.66   Mean   : 79.7   Mean   :14.09   Mean   :22.74  
##  3rd Qu.: 85.00   3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00  
##  Max.   :103.00   Max.   :100.0   Max.   :39.80   Max.   :64.00  
##      Expend         Grad.Rate    
##  Min.   : 3186   Min.   : 10.00  
##  1st Qu.: 6751   1st Qu.: 53.00  
##  Median : 8377   Median : 65.00  
##  Mean   : 9660   Mean   : 65.46  
```

```
##  3rd Qu.:10830   3rd Qu.: 78.00
##  Max.   :56233   Max.   :118.00
```

```r
head(college)
```

```
##   Private Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
## 1     Yes 1660   1232    721        23        52        2885         537
## 2     Yes 2186   1924    512        16        29        2683        1227
## 3     Yes 1428   1097    336        22        50        1036          99
## 4     Yes  417    349    137        60        89         510          63
## 5     Yes  193    146     55        16        44         249         869
## 6     Yes  587    479    158        38        62         678          41
##   Outstate Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend
## 1     7440       3300   450     2200  70       78      18.1          12   7041
## 2    12280       6450   750     1500  29       30      12.2          16  10527
## 3    11250       3750   400     1165  53       66      12.9          30   8735
## 4    12960       5450   450      875  92       97       7.7          37  19016
## 5     7560       4120   800     1500  76       72      11.9           2  10922
## 6    13500       3335   500      675  67       73       9.4          11   9727
##   Grad.Rate
## 1        60
## 2        56
## 3        54
## 4        59
## 5        15
## 6        55
```

```r
# matrix scatterplot
par(mfrow = c(4, 5), mar = c(4, 4, 2, 1))

pairs(college[, 2:10], main = "Scatterplot")
```

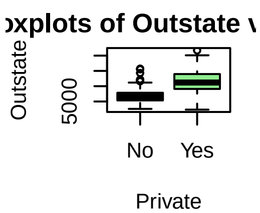



```r
# side by side plot
par(mfrow = c(4, 5), mar = c(4, 4, 2, 1))

boxplot(Outstate ~ Private, data = college, main = "Boxplots of Outstate v Private", xlab = "Private", y

# Add Elite
Elite = rep("No", nrow(college))
Elite[college$Top10perc>50]="Yes"
Elite = as.factor(Elite)
college = data.frame(college, Elite)
summary(college)
```

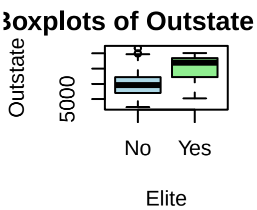
```
##    Private               Apps           Accept           Enroll
##  Length:777         Min.   :   81   Min.   :   72   Min.   :  35
##  Class :character   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242
##  Mode  :character   Median : 1558   Median : 1110   Median : 434
##                     Mean   : 3002   Mean   : 2019   Mean   : 780
##                     3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902
##                     Max.   :48094   Max.   :26330   Max.   :6392
##    Top10perc        Top25perc       F.Undergrad      P.Undergrad
##  Min.   : 1.00   Min.   :  9.0   Min.   :  139   Min.   :    1.0
##  1st Qu.:15.00   1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0
##  Median :23.00   Median : 54.0   Median : 1707   Median :  353.0
##  Mean   :27.56   Mean   : 55.8   Mean   : 3700   Mean   :  855.3
##  3rd Qu.:35.00   3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0
```

```
##  Max.   :96.00   Max.   :100.0   Max.   :31643   Max.   :21836.0
##     Outstate       Room.Board       Books           Personal
##  Min.   : 2340   Min.   :1780   Min.   :  96.0   Min.   : 250
##  1st Qu.: 7320   1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850
##  Median : 9990   Median :4200   Median : 500.0   Median :1200
##  Mean   :10441   Mean   :4358   Mean   : 549.4   Mean   :1341
##  3rd Qu.:12925   3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700
##  Max.   :21700   Max.   :8124   Max.   :2340.0   Max.   :6800
##       PhD            Terminal       S.F.Ratio      perc.alumni
##  Min.   :  8.00   Min.   : 24.0   Min.   : 2.50   Min.   : 0.00
##  1st Qu.: 62.00   1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00
##  Median : 75.00   Median : 82.0   Median :13.60   Median :21.00
##  Mean   : 72.66   Mean   : 79.7   Mean   :14.09   Mean   :22.74
##  3rd Qu.: 85.00   3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00
##  Max.   :103.00   Max.   :100.0   Max.   :39.80   Max.   :64.00
##      Expend         Grad.Rate      Elite
##  Min.   : 3186   Min.   : 10.00   No :699
##  1st Qu.: 6751   1st Qu.: 53.00   Yes: 78
##  Median : 8377   Median : 65.00
##  Mean   : 9660   Mean   : 65.46
##  3rd Qu.:10830   3rd Qu.: 78.00
##  Max.   :56233   Max.   :118.00
```

```r
# side by side plots
par(mfrow = c(4, 5), mar = c(4, 4, 2, 1))
```



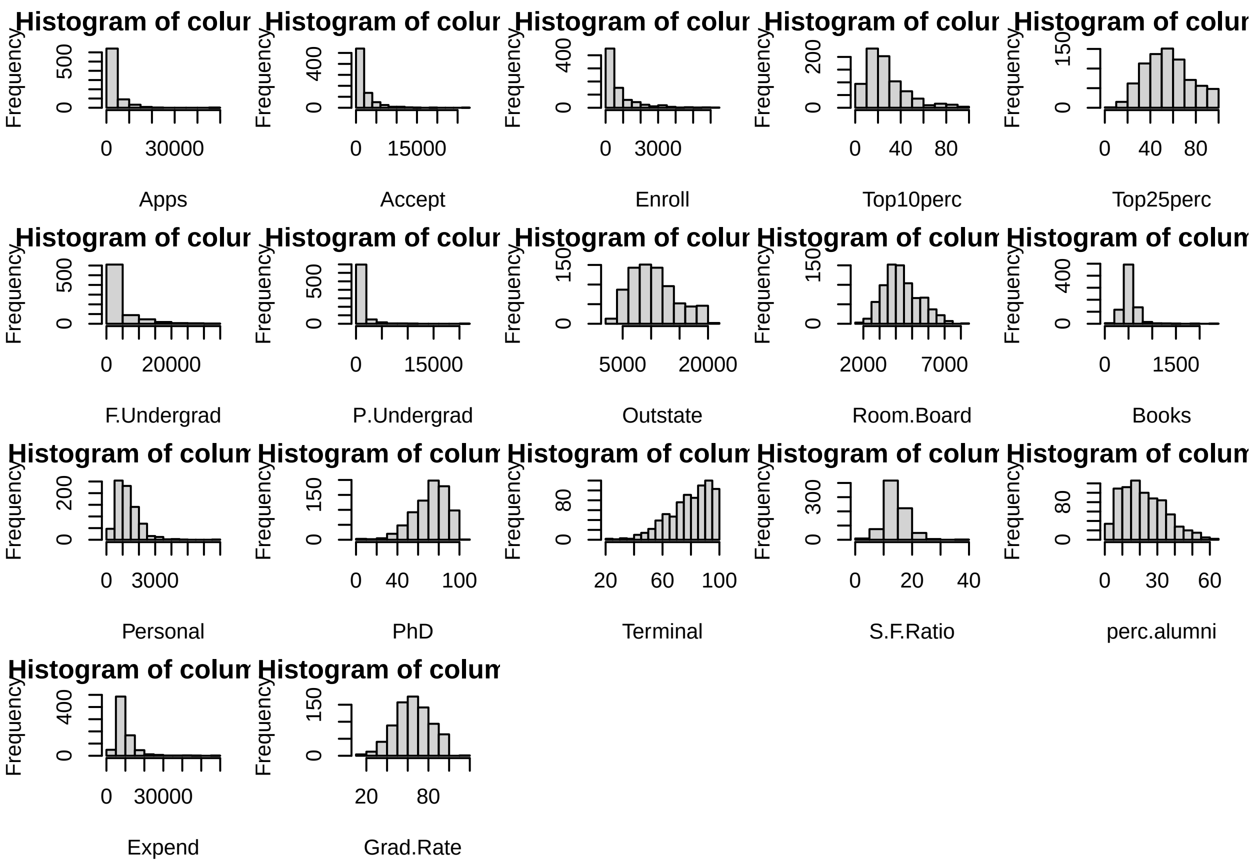
```r
boxplot(Outstate ~ Elite, data = college, main = "Boxplots of Outstate v Elite", xlab = "Elite", ylab =

# histograms
par(mfrow = c(4, 5), mar = c(4, 4, 2, 1))
```




```r
numeric_cols <- which(sapply(college, is.numeric))

print(numeric_cols)
```

```
##        Apps      Accept      Enroll   Top10perc   Top25perc F.Undergrad
##           2           3           4           5           6           7
## P.Undergrad    Outstate  Room.Board       Books    Personal         PhD
##           8           9          10          11          12          13
##    Terminal   S.F.Ratio perc.alumni      Expend   Grad.Rate
##          14          15          16          17          18
```

```r
for (col in numeric_cols) {
  hist(college[, col], main = paste("Histogram of column ", col), xlab = names(college)[col])
}
```



9.

```r
auto <- read.csv("C:/Users/chuan_71/OneDrive/Desktop/TRADING/ISL/Auto.csv")

head(auto)
```

```
##   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
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```r
# remove nan
auto <- na.omit(auto)
head(auto)
```

```
##   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
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```r
# quant vs qual predictors
quant_cols <- which(sapply(auto, is.numeric))
qual_cols <- which(!sapply(auto, is.numeric))
print(quant_cols)
```

```
##          mpg    cylinders displacement       weight acceleration         year
##            1            2            3            5            6            7
##       origin
##            8
```

```r
print(qual_cols)
```

```
## horsepower       name
##          4          9
```

```r
# function applicator function

app <- function(col_name, func){
  # init matrix
  results <- matrix(nrow = length(col_name), ncol = 3)
  rownames(results) <- col_name
  colnames(results) <- c("Mean", "Sd", "Range")

  # Loop
  for(i in seq_along(col_name)){
    col <- col_name[i]
    col_data <- auto[[col]]

    x <- mean(col_data)
    y <- sd(col_data)
    z <- paste(range(col_data), collapse = ", ")

    results[i, "Mean"] <- x
    results[i, "Sd"] <- y
    results[i, "Range"] <- z
  }
```

```r
  # return result
  return(results)
}

# mean, sd, range
app(quant_cols)
```

```
##   Mean                Sd                  Range
## 1 "23.5158690176322" "7.82580392894656"  "9, 46.6"
## 2 "5.45843828715365" "1.70157698079185"  "3, 8"
## 3 "193.53274559194"  "104.37958329993"   "68, 455"
## 5 "2970.26196473552" "847.904119489725"  "1613, 5140"
## 6 "15.5556675062972" "2.74999529297615"  "8, 24.8"
## 7 "75.9949622166247" "3.69000490146168"  "70, 82"
## 8 "1.57430730478589" "0.802549495797039" "1, 3"
```

```r
# remove 10th and 85th
auto <- auto[-c(10, 85), ]
new_quant_cols <- which(sapply(auto, is.numeric))
new_qual_cols <- which(!sapply(auto, is.numeric))

# range, mean, std
app(new_quant_cols)
```

```
##   Mean                Sd                  Range
## 1 "23.5286075949367" "7.83192521828543"  "9, 46.6"
## 2 "5.45569620253165" "1.69948834009909"  "3, 8"
## 3 "193.279746835443" "104.061131705913"  "68, 455"
## 5 "2970.23797468354" "847.764300207639"  "1613, 5140"
## 6 "15.5711392405063" "2.73349731700723"  "8, 24.8"
## 7 "76.020253164557"  "3.68142475960302"  "70, 82"
## 8 "1.57215189873418" "0.800846111706057" "1, 3"
```

10.