SML 201 – Week 11

John D. Storey

Spring 2016

Contents

Statistics, ML, and Data Science

Statistics

Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data.

https://en.wikipedia.org/wiki/Statistics

Machine Learning

Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Machine learning is closely related to and often overlaps with computational statistics; a discipline which also focuses in prediction-making through the use of computers.

https://en.wikipedia.org/wiki/Machine_learning

Data Science

Data Science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is a continuation of some of the data analysis fields such as statistics, data mining, and predictive analytics.

https://en.wikipedia.org/wiki/Data_science

Learning

A Definition

Statistical learning (or statistical machine learning) is largely about using statistical modeling ideas to solve machine learning problems.

"Learning" basically means using data to build or fit models.

Quotations

From *An Introduction to Statistical Learning*:

"Statistical learning refers to a vast set of tools for understanding data."

"Though the term statistical learning is fairly new, many of the concepts that underlie the field were developed long ago."

"Inspired by the advent of machine learning and other disciplines, statistical learning has emerged as a new subfield in statistics, focused on supervised and unsupervised modeling and prediction."

A Modeling Framework

Ordinary Least Squares

Suppose we observe data $(x_{11}, x_{21}, \ldots, x_{d1}, y_1), \ldots, (x_{1n}, x_{2n}, \ldots, x_{dn}, y_n)$. We have a response variable y_i and d explanatory variables $(x_{1i}, x_{2i}, \ldots, x_{di})$ per unit of observation.

Ordinary least squares models the variation of y in terms of $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_d x_d$.

OLS Model

The assumed model is

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_d X_{di} + E_i$$

where $\mathrm{E}[E_i] = 0$, $\mathrm{Var}(E_i) = \sigma^2$, and $\rho_{E_i,E_j} = 0$ for all $1 \leq i, j \leq n$ and $i \neq j$.

A More General Model

Let's collapse $X_i = (X_{1i}, X_{2i}, \ldots, X_{di})$. A more general model is

$$Y_i = f(X_i) + E_i,$$

with the same assumptions on E_i, for some function f that maps the d variables into the real numbers.

Modeling Fitting

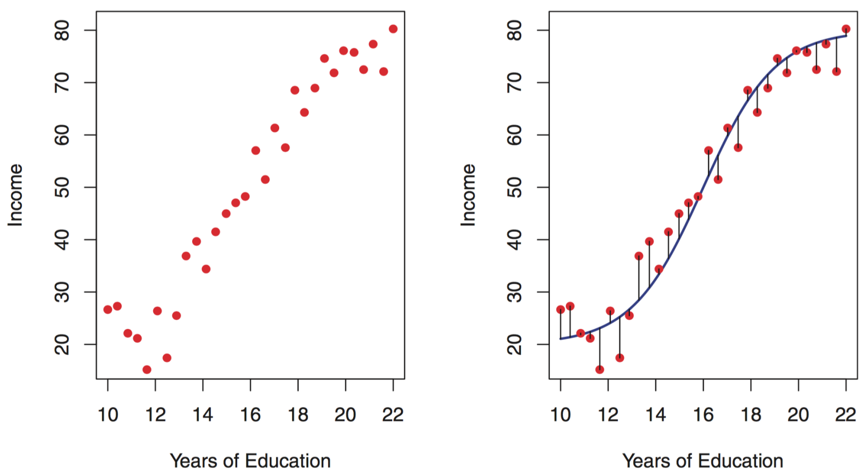


Figure credit: *ISL*

Example True Model

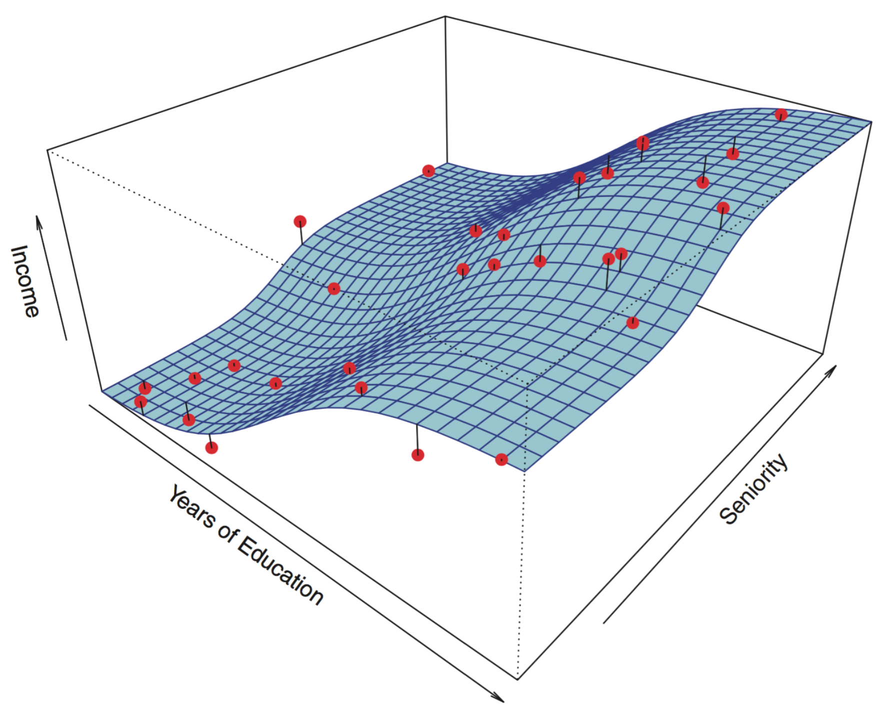


Figure credit: *ISL*

OLS Linear Model

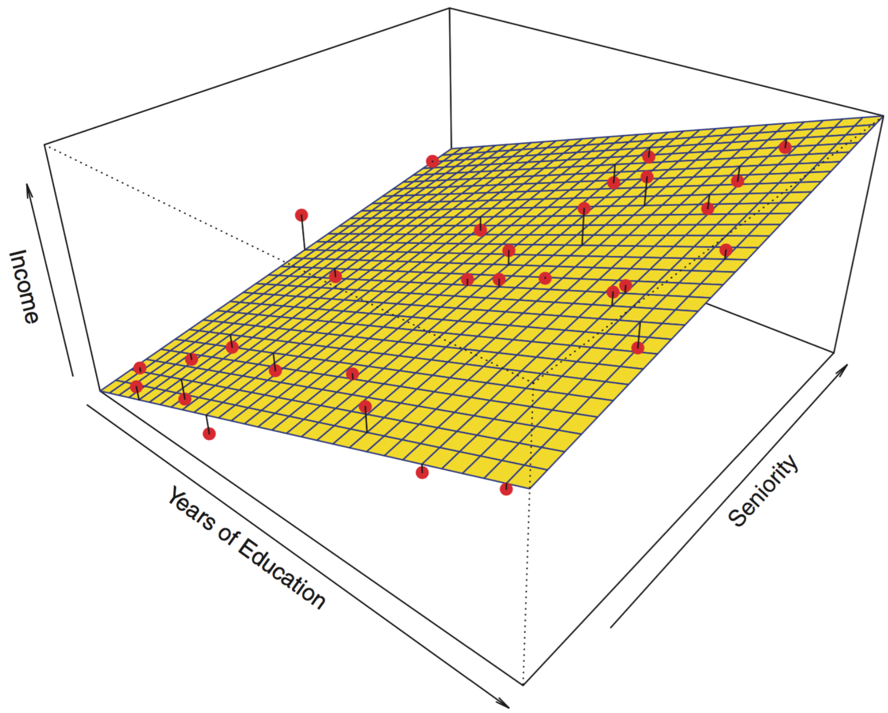


Figure credit: *ISL*

A Flexible Model

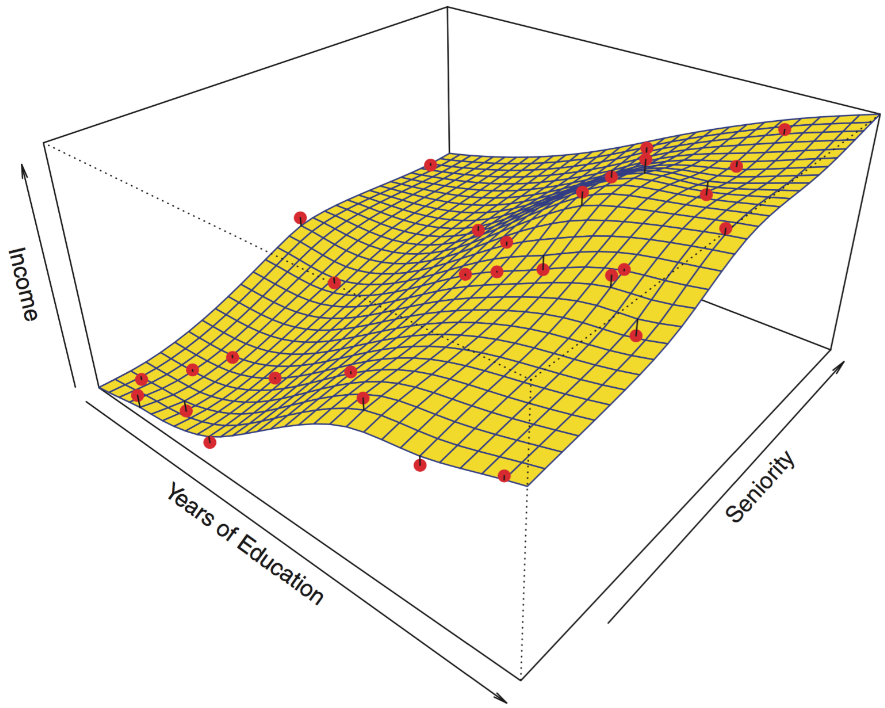


Figure credit: *ISL*

Variable Names

Input variables $(X_1, X_2, \ldots, X_d)$:

- explanatory variables
- covariates
- predictors
- independent variables
- feature variables

Output variable (Y):

- response variable
- dependent variable
- label
- outcome variable

Learning Types

Supervised learning is aimed at fitting models to (X, Y) so that we can model the output Y given the input X, typically on future observations. **Prediction models** are built by supervised learning.

Unsupervised learning (next week's topic) is aimed at fitting models to X alone to charcaterize the distribution of or find patterns in X.

Prediction

We often want to fit $Y = f(X) + E$ for either **prediction** or **inference**.

When observed x are readily available but y is not, the goal is usually *prediction*. If $\hat{f}(x)$ is the estimated model, we predict $\hat{y} = \hat{f}(x)$ for an observed x. Here, $\hat{f}$ is often treated as a black box and we mostly care that it provides accurate predictions.

Inference

When we co-observe x and y, we are often interested in understanding the way that y is explained by varying x or is a causal effect of x – and we want to be able to explicitly quantify these relationships. This is the goal of *inference*. Here, we want to be able to estimate and interpret f as accurately as possible – and have it be as close as possible to the underlying real-world mechanism connecting x to y.

Regression vs Classification

When $Y \in (-\infty, \infty)$, learning $Y = f(X) + E$ is called **regression**.

When $Y \in \{0, 1\}$ or more generally $Y \in \{c_1, c_2, \ldots, c_K\}$, we want to learn a function $f(X)$ that takes values in $\{c_1, c_2, \ldots, c_K\}$ so that $\Pr(Y = f(X))$ is as large as possible. This is called **classification**.

Parametric vs Nonparametric

A **parametric** model is a pre-specified form of $f(X)$ whose terms can be characterized by a formula and interpreted. This usually involves parameters on which inference can be performed, such as coefficients in the OLS model.

A **nonparametric** model is a data-driven form of $f(X)$ that is often very flexible and is not easily expressed or intepreted. A nonparametric model often does not include parameters on which we can do inference.

Accuracy of Learners

Decomposing Error

Let $\hat{Y} = \hat{f}(X)$ be the output of the learned model. Suppose that $\hat{f}$ and X are fixed. We can then define the error of this fitted model by:

$$\begin{aligned} \mathrm{E}\left[\left(Y - \hat{Y}\right)^2\right] &= \mathrm{E}\left[\left(f(X) + E - \hat{f}(X)\right)^2\right] && (1) \\ &= \mathrm{E}\left[\left(f(X) - \hat{f}(X)\right)^2\right] + \mathrm{Var}(E) && (2) \end{aligned}$$

The term $\mathrm{E}\left[\left(f(X) - \hat{f}(X)\right)^2\right]$ is the **reducible error** and the term $\mathrm{Var}(E)$ is the **irreducible error**.

Error Rates

On an observed data set $(x_1, y_1), \ldots, (x_n, y_n)$ we usually calculate error rates as follows.

For regression, we calculate the mean-squared error:

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i)\right)^2.$$

For classification, we calculate the misclassification rate:

$$\mathrm{MCR} = \frac{1}{n} \sum_{i=1}^{n} 1[y_i \neq \hat{f}(x_i)],$$

where $1[\cdot]$ is 0 or 1 whether the argument is false or true, respectively.

Training vs Testing

We typically fit the model on one data set and then assess its accuracy on an independent data set.

The data set used to fit the model is called the **training data set**.

The data set used to test the model is called the **testing data set** or **test data set**.

Important Questions

1. Why do we need training and testing data sets to accurately assess a learned model's accuracy?
2. How is this approach notably different from the inference approach we learned earlier?

Overfitting

Overfitting is a very important concept in statistical machine learning.

It occurs when the fitted model follows the noise term too closely.

In other words, when $\hat{f}(X)$ is overfitting the E term in $Y = f(X) + E$.

Performance of Different Models

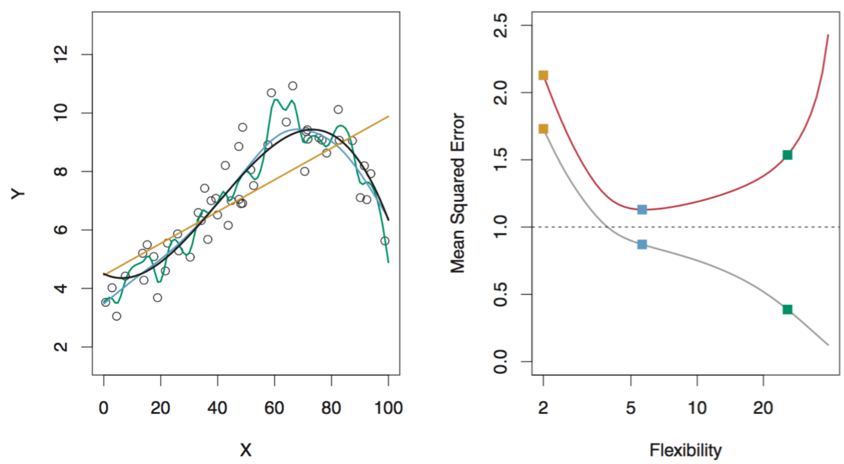


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

Figure credit: *ISL*

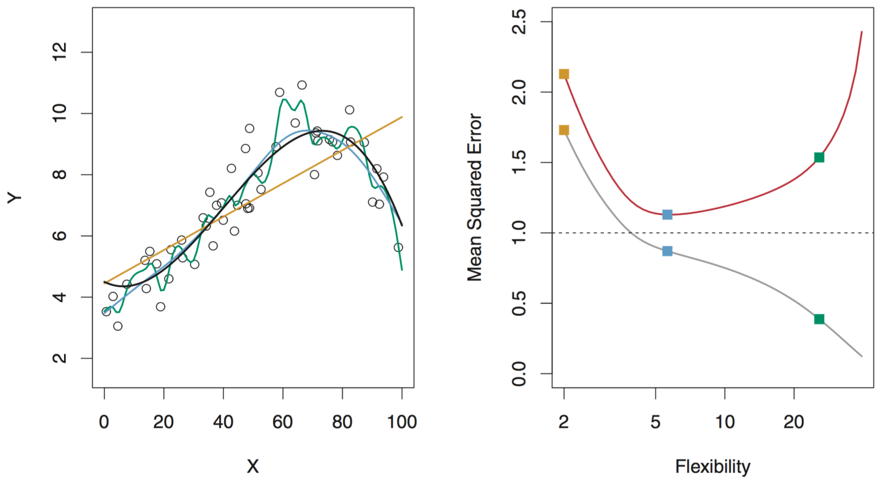


Figure credit: *ISL*

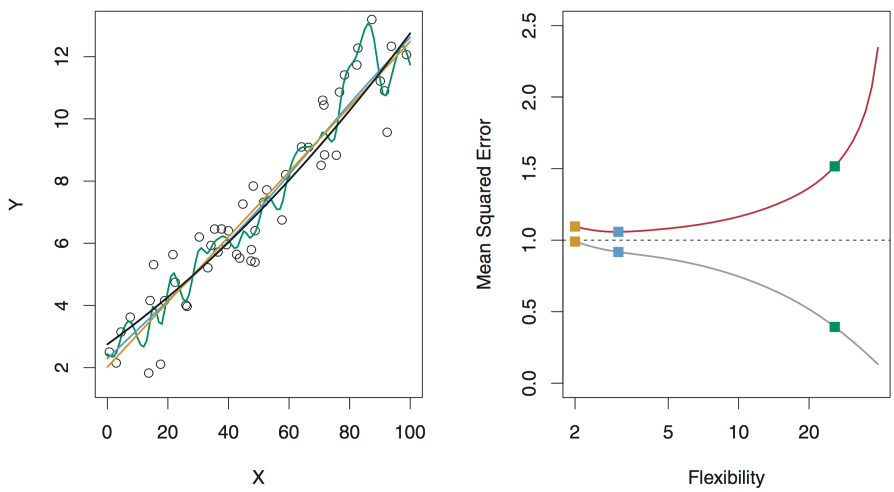


Figure credit: *ISL*

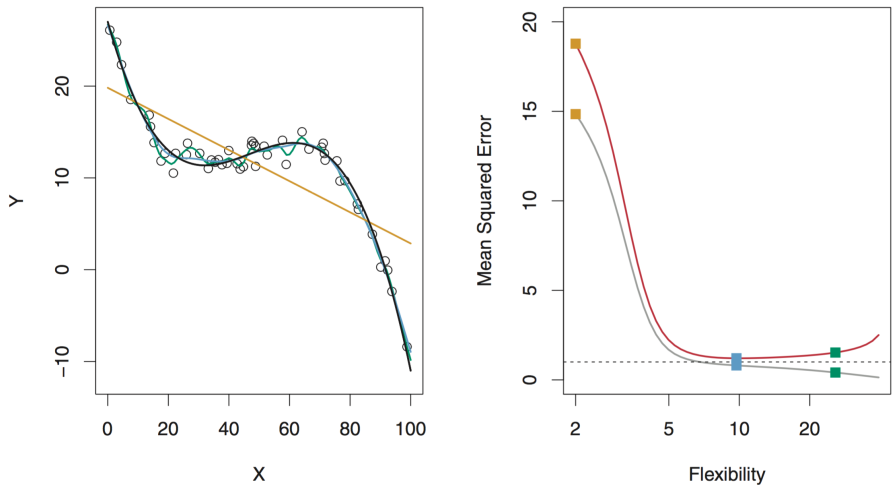


Figure credit: *ISL*

Trade-offs

Some Trade-offs

There are several important trade-offs encountered in prediction or learning:

- Bias vs variance
- Accuracy vs computational time
- Flexibility vs intepretability

These are not mutually exclusive phenomena.

Bias and Variance

$$
\begin{aligned}
\mathrm{E}\left[\left(Y-\hat{Y}\right)^2\right] &= \mathrm{E}\left[\left(f(X)+E-\hat{f}(X)\right)^2\right] && (3)\\
&= \mathrm{E}\left[\left(f(X)-\hat{f}(X)\right)^2\right]+\mathrm{Var}(E) && (4)\\
&= \left(f(X)-\mathrm{E}[\hat{f}(X)]\right)^2+\mathrm{Var}\left(\hat{f}(X)\right)^2+\mathrm{Var}(E) && (5)\\
&= \mathrm{bias}^2+\mathrm{variance}+\mathrm{Var}(E) && (6)
\end{aligned}
$$

Flexibility vs Interpretability

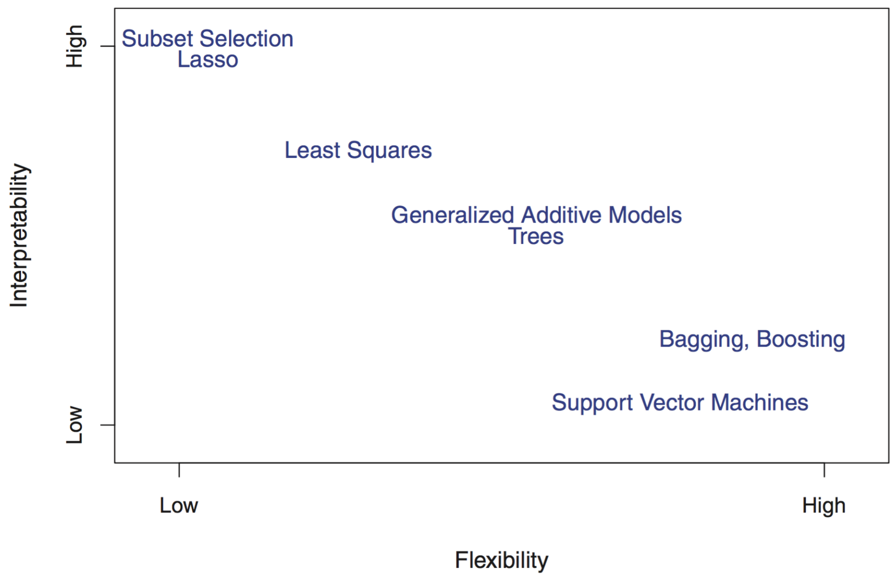


Figure credit: *ISL*

Logistic Regression

Definition

Logistic regression models Binomial distributed response variable in terms of linear combinations of explanatory variables.

Example: Grad School Admissions

```
> mydata <-
+   read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
> dim(mydata)
[1] 400   4
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
```

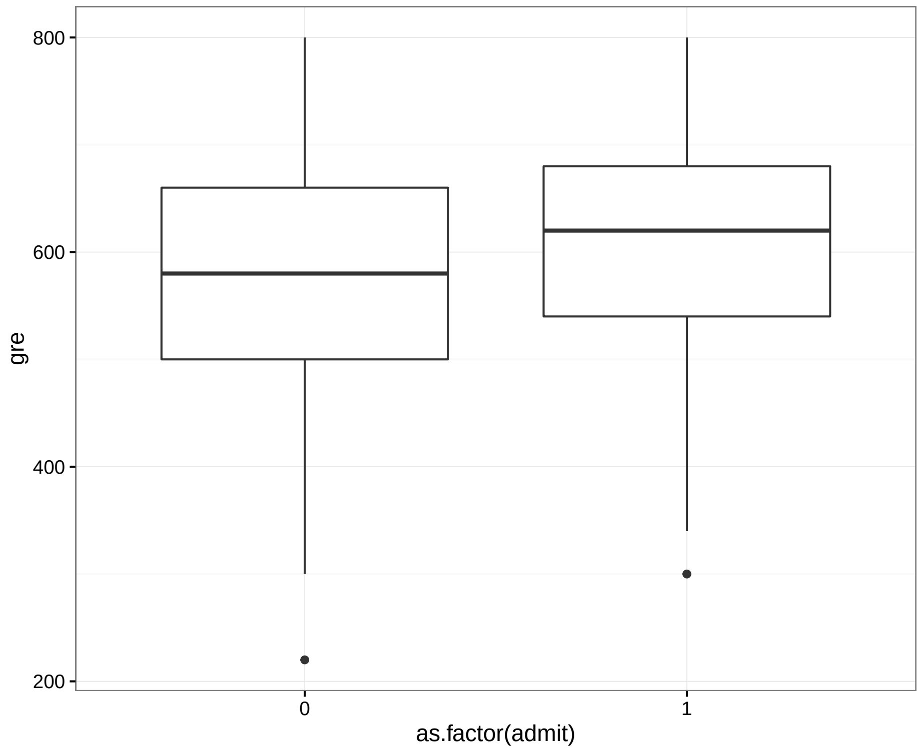
```
5     0 520 2.93    4
6     1 760 3.00    2
```

Data and analysis courtesy of http://www.ats.ucla.edu/stat/r/dae/logit.htm.

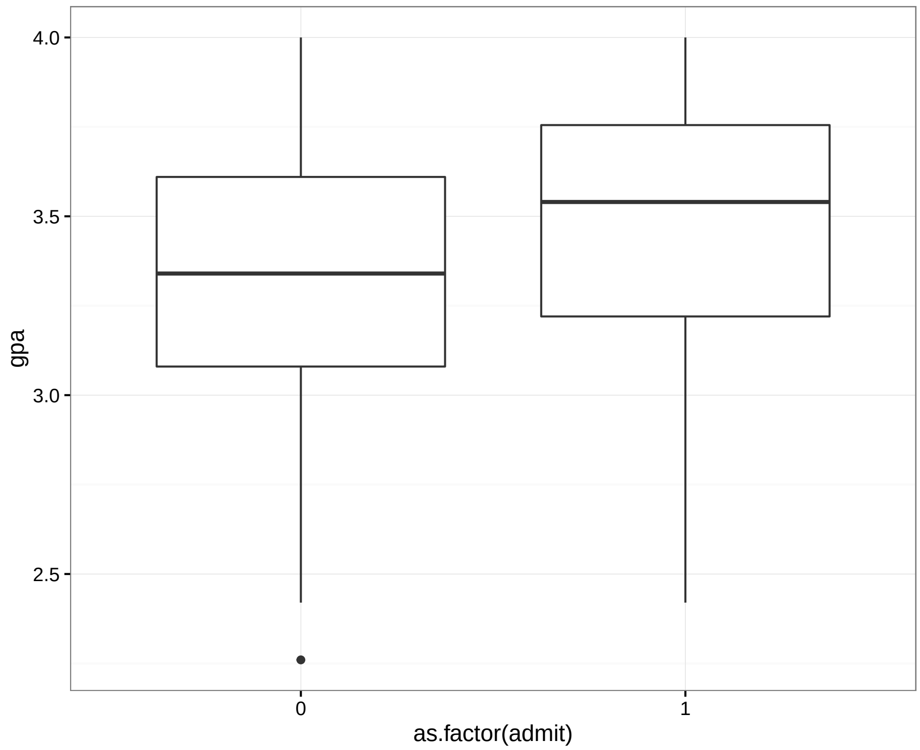
Explore the Data

```
> apply(mydata, 2, mean)
   admit      gre      gpa     rank
  0.3175 587.7000   3.3899   2.4850
> apply(mydata, 2, sd)
      admit         gre         gpa        rank
  0.4660867 115.5165364   0.3805668   0.9444602
>
> table(mydata$admit, mydata$rank)

     1  2  3  4
  0 28 97 93 55
  1 33 54 28 12
```

```
> ggplot(data=mydata) +
+   geom_boxplot(aes(x=as.factor(admit), y=gre))
```




```
> ggplot(data=mydata) +
+   geom_boxplot(aes(x=as.factor(admit), y=gpa))
```



The Model

Suppose we observe data $(x_{11}, x_{21}, \ldots, x_{d1}, y_1), \ldots, (x_{1n}, x_{2n}, \ldots, x_{dn}, y_n)$. We have a response variable y_i and d explanatory variables $(x_{1i}, x_{2i}, \ldots, x_{di})$ per unit of observation.

The assumption is that y is a realization from a $Y \sim \text{Binomia}(1, p)$ distribution where:

$$\log\left(\frac{p}{1-p}\right) = \log\left(\frac{\Pr(Y=1)}{\Pr(Y=0)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_d X_d$$

Logit Function

The logit function is

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

for $0 < p < 1$. The inverse logit function is

$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x}.$$

Fitting the Model

Logistic regression models are fit by finding the maximum likelihood estimate of p through and algorithm called iteratively reweighted least squares.

Logistic Regression in R

```
> mydata$rank <- factor(mydata$rank, levels=c(1, 2, 3, 4))
> myfit <- glm(admit ~ gre + gpa + rank,
+               data = mydata, family = "binomial")
> myfit

Call:  glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Coefficients:
(Intercept)          gre          gpa        rank2
  -3.989979     0.002264     0.804038    -0.675443
      rank3        rank4
  -1.340204    -1.551464

Degrees of Freedom: 399 Total (i.e. Null);  394 Residual
Null Deviance:      500
Residual Deviance: 458.5    AIC: 470.5
```

Summary of Fit

```
> summary(myfit)

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
```

ANOVA of Fit

```
> anova(myfit, test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: admit

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                   399     499.98
gre   1  13.9204       398     486.06 0.0001907 ***
gpa   1   5.7122       397     480.34 0.0168478 *
rank  3  21.8265       394     458.52 7.088e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example: Contraceptive Use

```
> cuse <-
+   read.table("http://data.princeton.edu/wws509/datasets/cuse.dat",
```

```
+               header=TRUE)
> dim(cuse)
[1] 16  5
> head(cuse)
    age education wantsMore notUsing using
1   <25       low       yes       53     6
2   <25       low        no       10     4
3   <25      high       yes      212    52
4   <25      high        no       50    10
5 25-29       low       yes       60    14
6 25-29       low        no       19    10
```

Data and analysis courtesy of http://data.princeton.edu/R/glms.html.

A Different Format

Note that in this data set there are multiple observations per explanatory variable configuration.

The last two columns of the data frame count the successes and failures per configuration.

```
> head(cuse)
    age education wantsMore notUsing using
1   <25       low       yes       53     6
2   <25       low        no       10     4
3   <25      high       yes      212    52
4   <25      high        no       50    10
5 25-29       low       yes       60    14
6 25-29       low        no       19    10
```

Fitting the Model

When this is the case, we call the `glm()` function slighlty differently.

```
> myfit <- glm(cbind(using, notUsing) ~ age + education +
+                 wantsMore, data=cuse, family = binomial)
> myfit

Call:  glm(formula = cbind(using, notUsing) ~ age + education + wantsMore,
    family = binomial, data = cuse)

Coefficients:
 (Intercept)      age25-29      age30-39      age40-49
```

```
     -0.8082        0.3894        0.9086        1.1892 
educationlow  wantsMoreyes 
     -0.3250       -0.8330 

Degrees of Freedom: 15 Total (i.e. Null);  10 Residual
Null Deviance:      165.8 
Residual Deviance: 29.92    AIC: 113.4
```

Summary of Fit

```
> summary(myfit)

Call:
glm(formula = cbind(using, notUsing) ~ age + education + wantsMore,
    family = binomial, data = cuse)

Deviance Residuals:
    Min       1Q   Median       3Q      Max  
-2.5148  -0.9376   0.2408   0.9822   1.7333  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)   -0.8082     0.1590  -5.083 3.71e-07 ***
age25-29       0.3894     0.1759   2.214  0.02681 *  
age30-39       0.9086     0.1646   5.519 3.40e-08 ***
age40-49       1.1892     0.2144   5.546 2.92e-08 ***
educationlow  -0.3250     0.1240  -2.620  0.00879 ** 
wantsMoreyes  -0.8330     0.1175  -7.091 1.33e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 165.772  on 15  degrees of freedom
Residual deviance:  29.917  on 10  degrees of freedom
AIC: 113.43

Number of Fisher Scoring iterations: 4
```

ANOVA of Fit

```
> anova(myfit, test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: cbind(using, notUsing)

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         15    165.772
age        3   79.192        12     86.581 < 2.2e-16 ***
education  1    6.162        11     80.418   0.01305 *
wantsMore  1   50.501        10     29.917 1.191e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

More on this Data Set

See http://data.princeton.edu/R/glms.html for more on fitting logistic regression to this data set.

A number of interesting choices are made that reveal more about the data and the ways that logistic regression can be utilized.

Spam Example

The Data

We will analyze a data set for determining whether an email is spam or not. The data can be found here, and it is also available in the kernal R package.

```
> library("dplyr")
> library("kernlab")
Warning: package 'kernlab' was built under R version 3.2.4
> library("broom")
> data("spam")
> spam <- tbl_df(spam)
> names(spam)
 [1] "make"              "address"
 [3] "all"               "num3d"
 [5] "our"               "over"
 [7] "remove"            "internet"
```

```
 [9] "order"            "mail"
[11] "receive"          "will"
[13] "people"           "report"
[15] "addresses"        "free"
[17] "business"         "email"
[19] "you"              "credit"
[21] "your"             "font"
[23] "num000"           "money"
[25] "hp"               "hpl"
[27] "george"           "num650"
[29] "lab"              "labs"
[31] "telnet"           "num857"
[33] "data"             "num415"
[35] "num85"            "technology"
[37] "num1999"          "parts"
[39] "pm"               "direct"
[41] "cs"               "meeting"
[43] "original"         "project"
[45] "re"               "edu"
[47] "table"            "conference"
[49] "charSemicolon"    "charRoundbracket"
[51] "charSquarebracket" "charExclamation"
[53] "charDollar"       "charHash"
[55] "capitalAve"       "capitalLong"
[57] "capitalTotal"     "type"
```

Contents of the Data Set

```
> dim(spam)
[1] 4601   58
> head(spam)
Source: local data frame [6 x 58]

   make address   all num3d   our  over remove internet order
  (dbl)   (dbl) (dbl) (dbl) (dbl) (dbl)  (dbl)    (dbl) (dbl)
1  0.00    0.64  0.64     0  0.32  0.00   0.00     0.00  0.00
2  0.21    0.28  0.50     0  0.14  0.28   0.21     0.07  0.00
3  0.06    0.00  0.71     0  1.23  0.19   0.19     0.12  0.64
4  0.00    0.00  0.00     0  0.63  0.00   0.31     0.63  0.31
5  0.00    0.00  0.00     0  0.63  0.00   0.31     0.63  0.31
6  0.00    0.00  0.00     0  1.85  0.00   0.00     1.85  0.00
Variables not shown: mail (dbl), receive (dbl), will (dbl),
  people (dbl), report (dbl), addresses (dbl), free (dbl),
```

```
  business (dbl), email (dbl), you (dbl), credit (dbl), your
  (dbl), font (dbl), num000 (dbl), money (dbl), hp (dbl), hpl
  (dbl), george (dbl), num650 (dbl), lab (dbl), labs (dbl),
  telnet (dbl), num857 (dbl), data (dbl), num415 (dbl), num85
  (dbl), technology (dbl), num1999 (dbl), parts (dbl), pm
  (dbl), direct (dbl), cs (dbl), meeting (dbl), original
  (dbl), project (dbl), re (dbl), edu (dbl), table (dbl),
  conference (dbl), charSemicolon (dbl), charRoundbracket
  (dbl), charSquarebracket (dbl), charExclamation (dbl),
  charDollar (dbl), charHash (dbl), capitalAve (dbl),
  capitalLong (dbl), capitalTotal (dbl), type (fctr)
```

Convert the Response Variable

```
> spam <- spam %>%
+         mutate(response=as.numeric(type=="spam")) %>%
+         select(-type)
> mean(spam$response)
[1] 0.3940448
```

Logistic Regression: 1 Variable

```
> myfit <- glm(response ~ edu, family=binomial, data=spam)
Warning: glm.fit: fitted probabilities numerically 0 or 1
occurred
> myfit

Call:  glm(formula = response ~ edu, family = binomial, data = spam)

Coefficients:
(Intercept)          edu
    -0.2987      -2.2198

Degrees of Freedom: 4600 Total (i.e. Null);  4599 Residual
Null Deviance:      6170
Residual Deviance: 5907     AIC: 5911
```

Summary

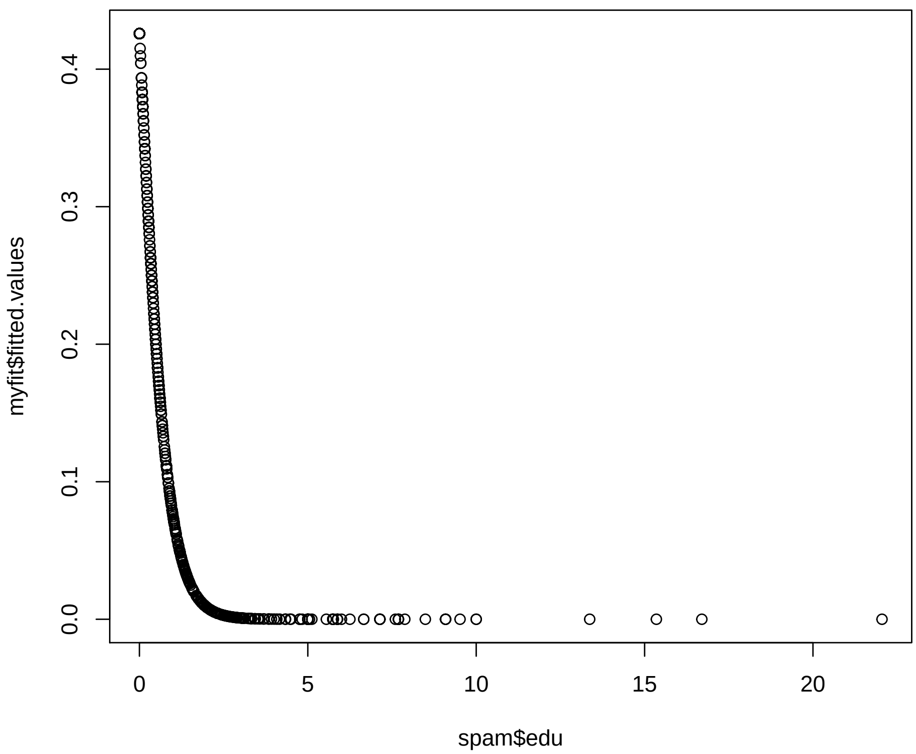
```
> summary(myfit)

Call:
glm(formula = response ~ edu, family = binomial, data = spam)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.054  -1.054  -1.054   1.307   3.567

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.29872    0.03128  -9.551   <2e-16 ***
edu         -2.21983    0.25482  -8.711   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6170.2  on 4600  degrees of freedom
Residual deviance: 5907.2  on 4599  degrees of freedom
AIC: 5911.2

Number of Fisher Scoring iterations: 7
```

Logit Function

```
> logit <- function(p, tol=1e-10) {
+   p <- pmin(pmax(tol, p), 1-tol)
+   log(p/(1-p))
+ }
```

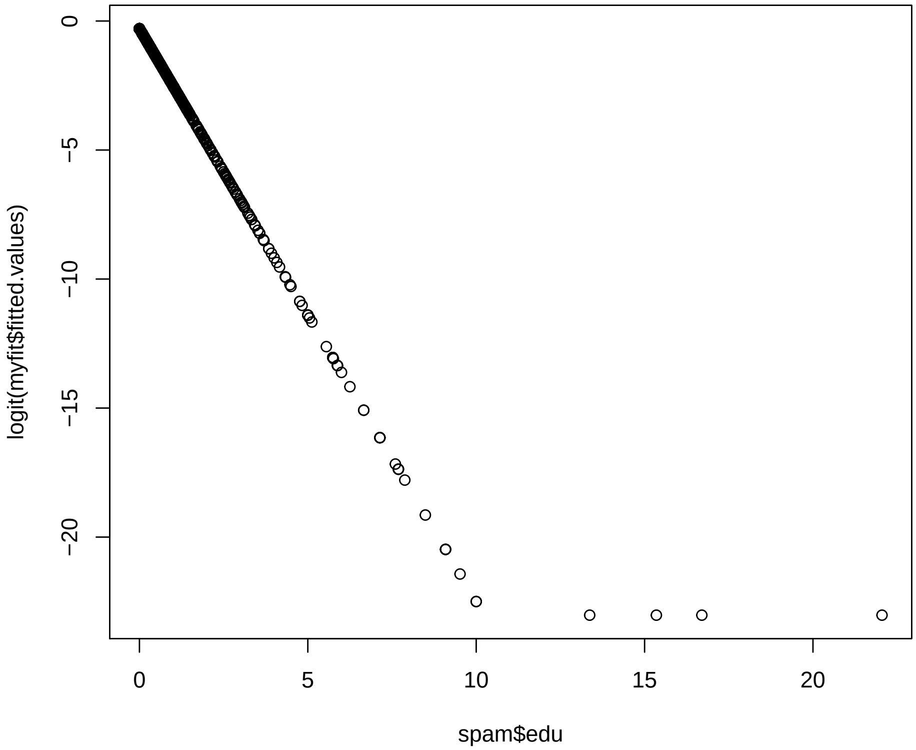
Visualization

```
> plot(spam$edu, myfit$fitted.values)
```



Visualization

```
> plot(spam$edu, logit(myfit$fitted.values))
```



Logistic Regression: All Variables

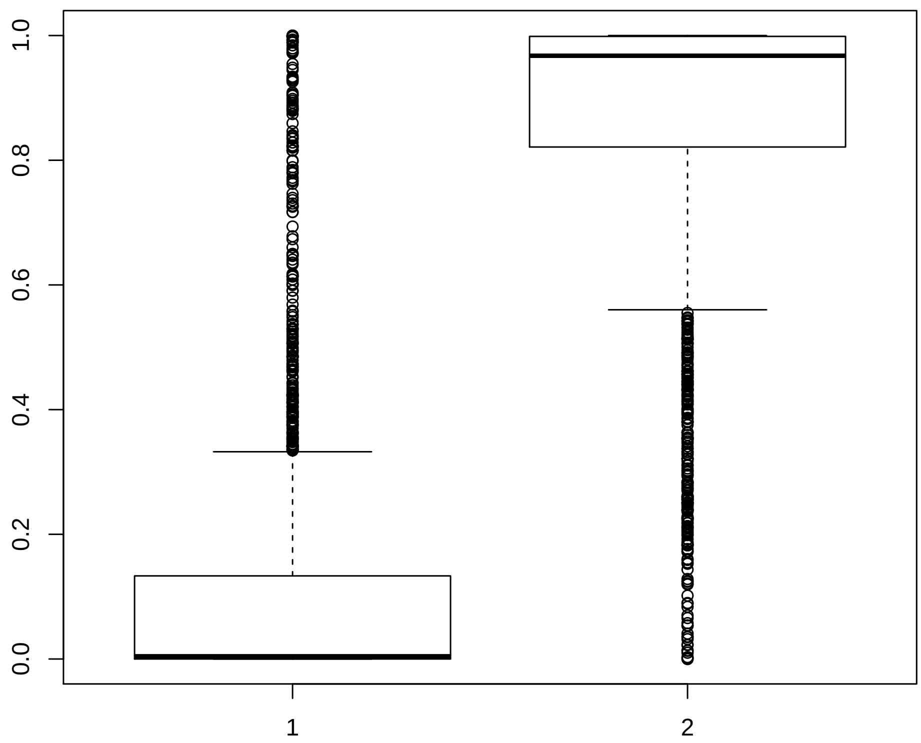
```
> myfit <- glm(response ~ ., family=binomial, data=spam)
Warning: glm.fit: fitted probabilities numerically 0 or 1
occurred
> x <- tidy(myfit)
> head(x)
         term   estimate std.error  statistic      p.value
1 (Intercept) -1.5686144 0.1420362 -11.043767 2.349719e-28
2        make -0.3895185 0.2314521  -1.682933 9.238799e-02
3     address -0.1457768 0.0692792  -2.104194 3.536157e-02
4         all  0.1141402 0.1103011   1.034806 3.007594e-01
5       num3d  2.2515195 1.5070099   1.494031 1.351675e-01
6         our  0.5623844 0.1017997   5.524423 3.305708e-08
```

Misclassification Rate (Biased)

```
> pred_response <- as.numeric(myfit$fitted.values >= 0.5)
> mean(pred_response == spam$response)
[1] 0.9313193
```

Probabilites on Full Data Fit

```
> boxplot(myfit$fitted.values[spam$response==0],
+         myfit$fitted.values[spam$response==1])
```



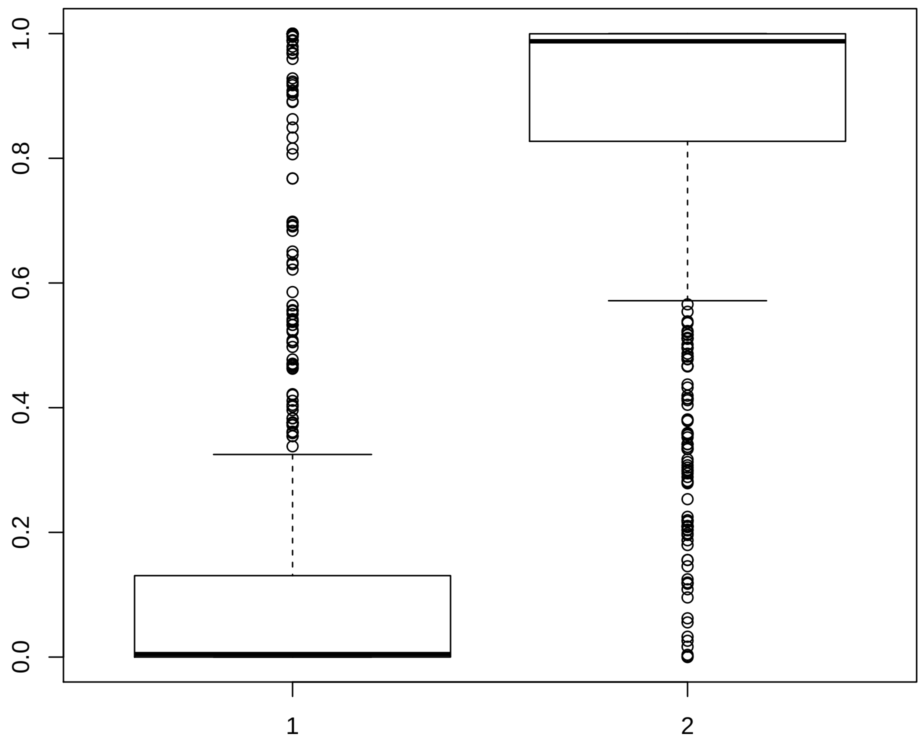
Train and Test Sets

```
> set.seed(210)
> v <- sample(nrow(spam), size=round(2*nrow(spam)/3))
> spam0 <- spam[v,]  ## training data
> spam1 <- spam[-v,]  ## test data
> fit0 <- glm(response ~ ., family=binomial, data=spam0)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1
occurred
> pred_prob <- predict(fit0, newdata=spam1[,-ncol(spam1)],
+                       type="response")
```

Trained Probabilities on Test Set

```
> boxplot(pred_prob[spam1$response==0],
+         pred_prob[spam1$response==1])
```



Misclassification Rate (Unbiased)

```
> pred_response <- as.numeric(pred_prob >= 0.5)
> mean(pred_response == spam1$response)
[1] 0.9132986
```

A Prediction Framework in R

The `caret` Package

R has several convenient frameworks for building and evaluating prediction models.

One of them is contained in the package `caret`.

We will go through an example, courtesy of this *RPubs* publication.

```
> library("caret")
> ## remove and reload data to undo my earlier changes
> rm(spam)
> data("spam", package="kernlab")
```

Set Up Training and Test Data

```
> set.seed(201)
> inTrain <- createDataPartition(y=spam$type, p=0.6,
+                                list=FALSE)
> training <-spam[inTrain,]
> testing <- spam[-inTrain,]
```

Fit Logistic Model

```
> modelFit_glm <- train(type ~., data=training, method="glm")
> modelFit_glm
Generalized Linear Model

2761 samples
  57 predictor
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2761, 2761, 2761, 2761, 2761, 2761, ...
Resampling results:

  Accuracy   Kappa
  0.9192593  0.8295914
```

Evaluate Logistic

```
> predictions <- predict(modelFit_glm, newdata=testing)
> confusionMatrix(predictions, testing$type)
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam    1068   93
   spam         47  632

               Accuracy : 0.9239
                 95% CI : (0.9108, 0.9356)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8389
 Mcnemar's Test P-Value : 0.0001428

            Sensitivity : 0.9578
            Specificity : 0.8717
         Pos Pred Value : 0.9199
         Neg Pred Value : 0.9308
             Prevalence : 0.6060
         Detection Rate : 0.5804
   Detection Prevalence : 0.6310
      Balanced Accuracy : 0.9148

       'Positive' Class : nonspam
```

Fit Support Vector Machine

```
> modelFit_svm <- train(type ~., data=training, method="svmLinear")
> modelFit_svm
Support Vector Machines with Linear Kernel

2761 samples
  57 predictor
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)
```

```
Summary of sample sizes: 2761, 2761, 2761, 2761, 2761, 2761, ...
Resampling results:

  Accuracy   Kappa
  0.9240032  0.8394918

Tuning parameter 'C' was held constant at a value of 1
```

Evaluate SVM

```
> predictions <- predict(modelFit_svm, newdata=testing)
> confusionMatrix(predictions, testing$type)
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam    1069   85
   spam         46  640

               Accuracy : 0.9288
                 95% CI : (0.9161, 0.9401)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8495
 Mcnemar's Test P-Value : 0.0008999

            Sensitivity : 0.9587
            Specificity : 0.8828
         Pos Pred Value : 0.9263
         Neg Pred Value : 0.9329
             Prevalence : 0.6060
         Detection Rate : 0.5810
   Detection Prevalence : 0.6272
      Balanced Accuracy : 0.9208

       'Positive' Class : nonspam
```

Fit Classification Tree Model

```
> modelFit_rpart <- train(type ~., data=training, method="rpart")
> modelFit_rpart
CART

2761 samples
  57 predictor
   2 classes: 'nonspam', 'spam'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2761, 2761, 2761, 2761, 2761, 2761, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.06893382  0.8271786  0.6344264
  0.06985294  0.8271786  0.6344264
  0.47794118  0.7454901  0.4357645

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was cp = 0.06985294.
```

Evaluate Classification Tree

```
> predictions <- predict(modelFit_rpart, newdata=testing)
> confusionMatrix(predictions, testing$type)
Confusion Matrix and Statistics

          Reference
Prediction nonspam spam
   nonspam     900  178
   spam        215  547

               Accuracy : 0.7864
                 95% CI : (0.767, 0.8049)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5567
 Mcnemar's Test P-Value : 0.06938
```

```
            Sensitivity : 0.8072
            Specificity : 0.7545
         Pos Pred Value : 0.8349
         Neg Pred Value : 0.7178
             Prevalence : 0.6060
         Detection Rate : 0.4891
   Detection Prevalence : 0.5859
      Balanced Accuracy : 0.7808

       'Positive' Class : nonspam
```

Extras

License

https://github.com/SML201/lectures/blob/master/LICENSE.md

Source Code

https://github.com/SML201/lectures/tree/master/week11

Session Information

```
> sessionInfo()
R version 3.2.3 (2015-12-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.4 (El Capitan)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
[1] kernlab_0.9-24  broom_0.4.0     dplyr_0.4.3
[4] ggplot2_2.1.0   knitr_1.12.3    magrittr_1.5
[7] devtools_1.10.0
```

```
loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4       mnormt_1.5-3      munsell_0.4.3
 [4] lattice_0.20-33   colorspace_1.2-6  R6_2.1.2
 [7] stringr_1.0.0     plyr_1.8.3        tools_3.2.3
[10] parallel_3.2.3    grid_3.2.3        gtable_0.2.0
[13] nlme_3.1-125      psych_1.5.8       DBI_0.3.1
[16] htmltools_0.3.5   lazyeval_0.1.10   yaml_2.1.13
[19] digest_0.6.9      assertthat_0.1    tidyr_0.4.1
[22] reshape2_1.4.1    formatR_1.3       memoise_1.0.0
[25] evaluate_0.8.3    rmarkdown_0.9.5.9 labeling_0.3
[28] stringi_1.0-1     scales_0.4.0
```