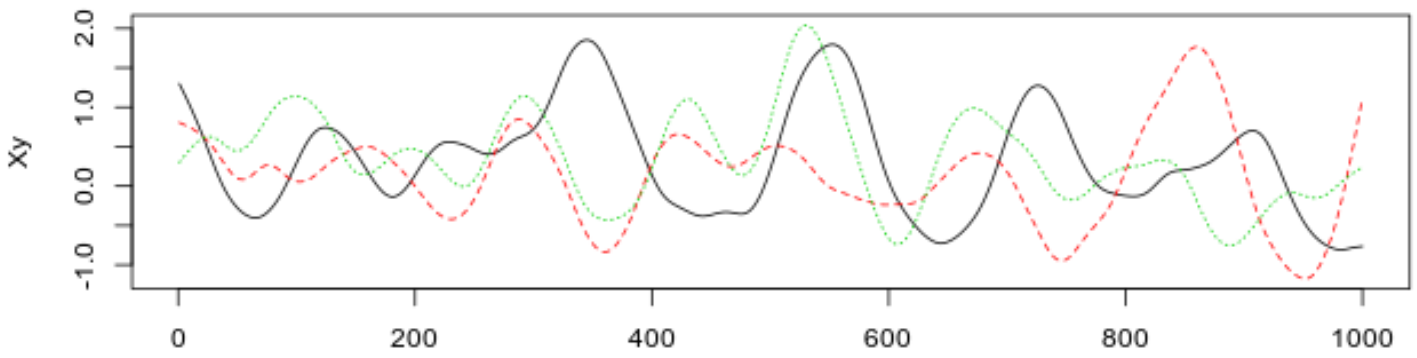
Chapter 5

Notes

```
load(paste0(here::here(), "/ISLR/5.R.RData"))

summary(lm(y ~ X1 + X2, data = Xy))
```

```

Call:
lm(formula = y ~ X1 + X2, data = Xy)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.44171 -0.25468 -0.01736  0.33081  1.45860 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.26583    0.01988  13.372  < 2e-16 ***
X1           0.14533    0.02593   5.604 2.71e-08 ***
X2           0.31337    0.02923  10.722  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5451 on 997 degrees of freedom
Multiple R-squared:  0.1171,    Adjusted R-squared:  0.1154 
F-statistic: 66.14 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(1,1))
matplot(Xy,type="l")
```




```
alpha = function(x,y){
  vx = var(x)
  vy = var(y)
  cxy= cov(x,y)
  (vy-cxy)/(vx+vy-2*cxy)
}
alpha(Xy$X1,Xy$y)
```

```
[1] 0.4167192
```

```
alpha.fn = function(data,index){
  with(data[index,],alpha(Xy$X1,Xy$y))
}

alpha.fn<-function(data, index) {
  fit1<-lm(y~., data=Xy[index,])
  coefficients(fit1)[['X1']]
}

set.seed(1)
alpha.fn(Xy, sample(1:100,100,replace=TRUE))
```

```
[1] 0.1967913
```

```
boot.out <- boot(Xy,alpha.fn,R=1000)

?tsboot

se_stat <- function(data, index) {
   fit1 <- lm(y ~., data = data[index,])
```

```
   coefficients(fit1)[['X1']]
}

tsboot.out <- tsboot(Xy, se_stat, R = 1000, l = 100, sim = "fixed")
tsboot.out
```

```
BLOCK BOOTSTRAP FOR TIME SERIES

Fixed Block Length of 100

Call:
tsboot(tseries = Xy, statistic = se_stat, R = 1000, l = 100,
    sim = "fixed")


Bootstrap Statistics :
     original      bias    std. error
t1* 0.1453263 -0.00322118   0.1939388
```

Cross-Validation

Cross-validation is the process of splitting the training data into multiple subsets that can be used for evaluating the out-of-sample performance of a statistical model.

There are multiple strategies for this technique.

The Validation Set approach

The most basic strategy for CV is a training/test split of the sample.

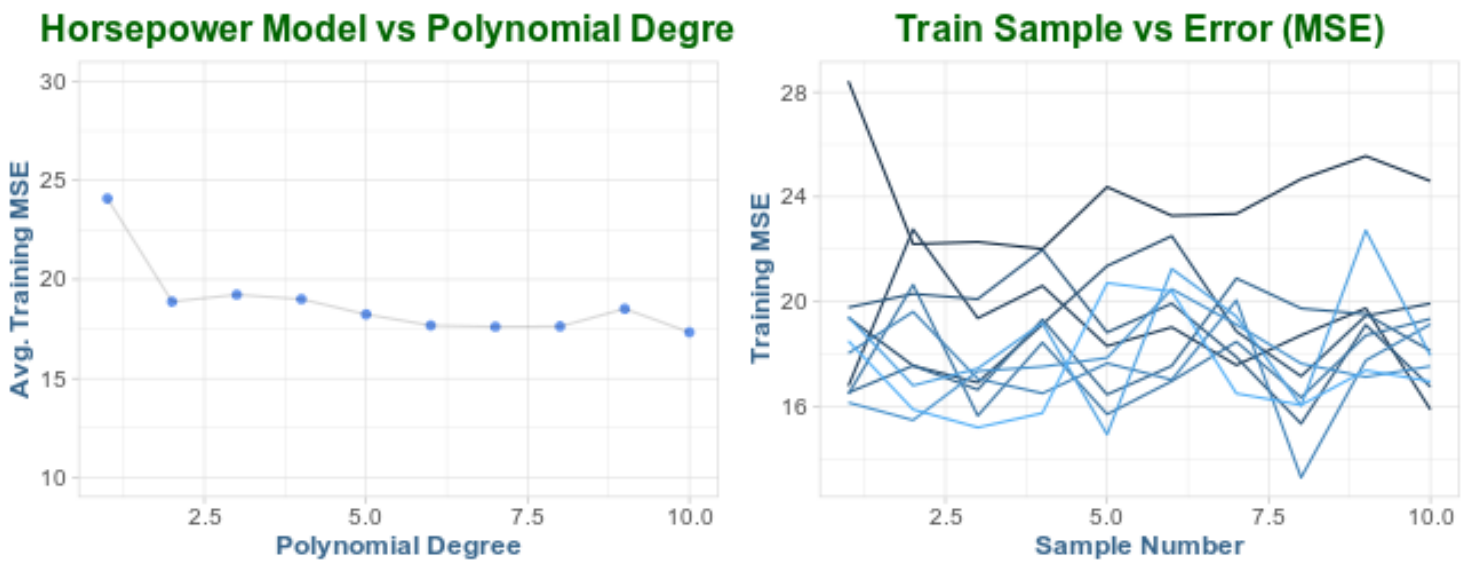
Additionally, stratification can be used to ensure even splitting when the response variable is unevenly distributed in the sample (low response logistic regression, for example).

Using Auto:

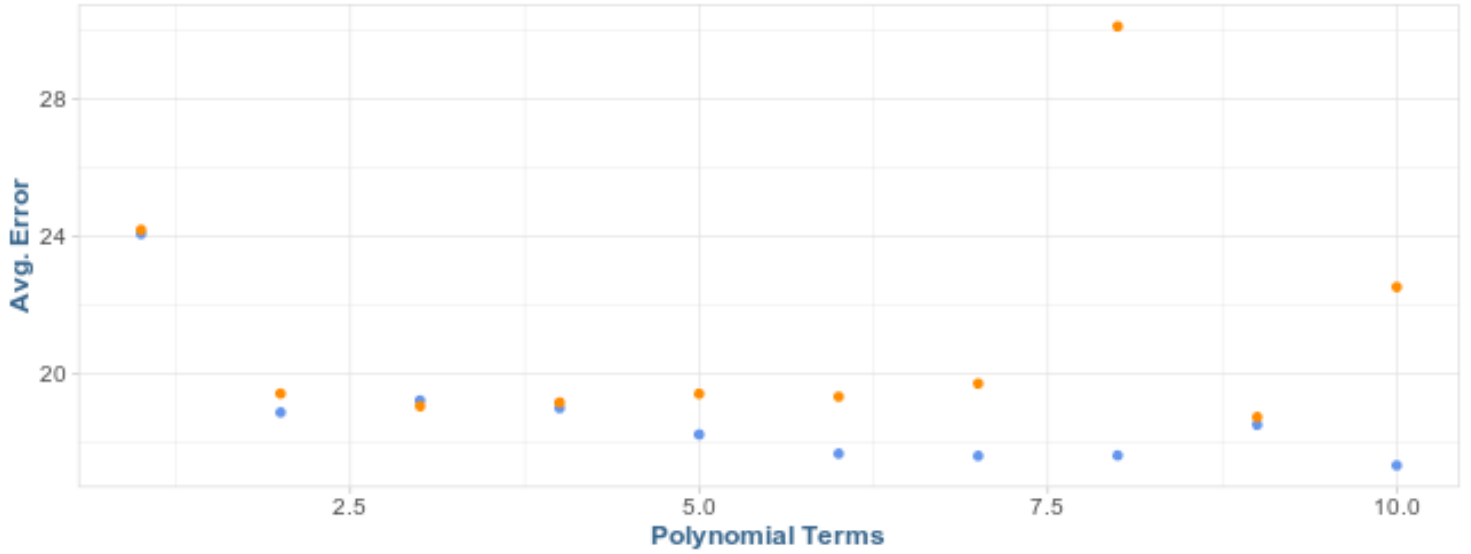
```
auto <- data.table(ISLR::Auto)
```

```
models <- list()

base.model <- "mpg ~ horsepower"

prev <- ""
for(term in 1:10)
{
   cur <- ifelse(term > 1, paste(prev, rep(paste0("+ I(horsepower^", term,")"))), "")
```

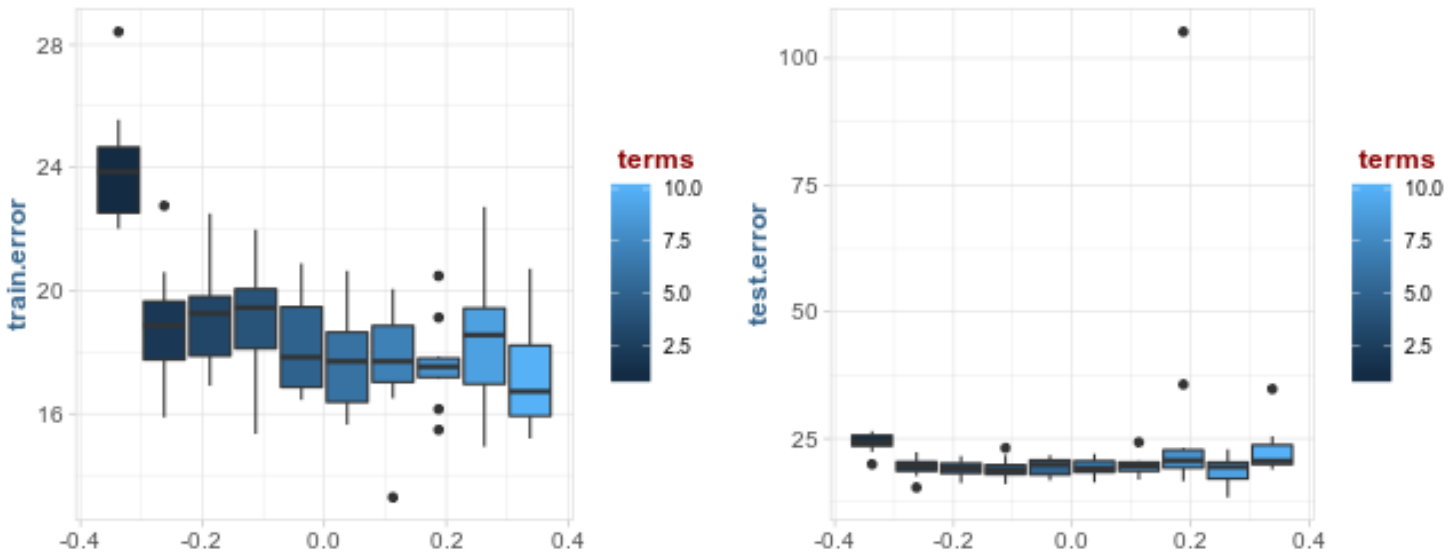
```
   fmla <- as.formula(paste0(base.model, cur))

   train.error <- numeric(10); test.error <- numeric(10)

   for(iter in 1:10)
   {
      auto.split <- initial_split(auto, prop = .5)

      auto.train <- training(auto.split)
      model <- lm(fmla, data = auto.train)
      train.error[iter] <- mean(model$residuals^2)

      auto.test <- testing(auto.split)
      auto.test$pred <- predict(model, newdata = auto.test)
      test.error[iter] <- with(auto.test, mean( (mpg - pred)^2 ) )
   }

   models[[term]] <- list(terms = term, train.error = train.error, test.error = test.error)
   prev <- cur
}

auto.fits <- rbindlist(models, fill = F)

p1 <- auto.fits %>%
   group_by(terms) %>%
   summarise(mean_error = mean(train.error)) %>%
   ggplot(., aes(terms, mean_error)) +
      geom_point(col = "cornflowerblue") +
      geom_line(alpha = .15) +
      labs(title = "Horsepower Model vs Polynomial Degree", x = "Polynomial Degree", y = "Avg.
      scale_y_continuous(limits = c(10, 30))

p2 <- auto.fits[, .(num = 1:.N, train.error), by = list(terms)] %>%
   ggplot() +
   geom_line(aes(num, train.error, group = terms, col = terms)) +
   labs(title = "Train Sample vs Error (MSE)", x = "Sample Number", y = "Training MSE") +
   theme(legend.position = "none")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```




```
ggplot(auto.fits[, .(train = mean(train.error), test = mean(test.error)), by = list(terms)]) +
   geom_point(aes(terms, train), col = "cornflowerblue") +
   geom_point(aes(terms, test), col = "darkorange") +
   labs(x = "Polynomial Terms", y = "Avg. Error")
```




```
p1 <- ggplot(auto.fits) +
   geom_boxplot(aes(y = train.error, group = terms, fill = terms))

p2 <- ggplot(auto.fits) +
   geom_boxplot(aes(y = test.error, group = terms, fill = terms))

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



Drawbacks:

- 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 in the validation set.
- 2.) In the validation approach, only a subset of the observations - those that are included in the training set rather than the validation set - are used to fit the model. This can lead to overestimation of model performance.

Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach, but attempts to address some of the drawbacks.

The basic premise of LOOCV is to leave exactly 1 observation out of the data to test against, and use the remaining n-1 observations to train the model.

After each resample, the final test error is produced by:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i$$

This is more of systematic approach to model training/validation.

Advantages:

- Far less bias than the validation/test set.

k-Fold Cross-Validation

An alternative to LOOCV is k-Fold CV. This approach randomly divides the set of observations into *k* groups, or *k*-folds, of approximately equal size.

The first fold is treated as a validation set, and the method is fit on the remaining *k-1* folds. LOOCV is a special case of k-Fold, (k = n).

However, there is a bias/variance trade-off associated with the choice of k. If you have a large number of k (example, n), then each model output will be highly correlated to each other.

Typically, a good choice of k is 5 or 10.

Cross-Validation on Classification

Instead of MSE for error metric, we will use:

$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} Err_i$

The Bootstrap

Bootstrapping is the process of resampling a data set (with replacement) to obtain confidence intervals (SE) for unknown population parameters.

R lab

The Validation Set Approach

```
set.seed(1)

train <- sample(392, 196)

lm.fit <- lm(mpg ~ horsepower, data = auto, subset = train)

mean(lm.fit$residuals^2) # train MSE
```

```
[1] 25.05959
```

```
test <- auto[!train]
test$pred <- predict(lm.fit, newdata = test)
with(test, mean((mpg - pred)^2)) # test MSE
```

```
[1] 23.26601
```

```
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = auto, subset = train)
test$pred <- predict(lm.fit2, newdata = test) # update predictions
with(test, mean((mpg - pred)^2)) # model 2 MSE
```

```
[1] 18.71646
```

```
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = auto, subset = train)
test$pred <- predict(lm.fit3, newdata = test) # update predictions
with(test, mean((mpg - pred)^2)) # model 2 MSE
```

```
[1] 18.79401
```

Leave-One-Out CV

```
glm.fit <- glm(mpg ~ horsepower, data = auto)
coef(glm.fit)
```

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

```
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(7)

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)$delta[1]
}

cv.error.10
```

```
 [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115
 [9] 19.06863 19.49093
```

The Bootstrap

```
portfolio <- data.table(ISLR::Portfolio)

alpha.fn <- function(data, index ) {
   X <- data$X; Y <- data$Y
   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] 0.5758321
```

```
boot(data = portfolio, statistic = 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           0
```

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, 1:396)
```

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

```
set.seed(1)

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

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

```
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  0.0549915227 0.841925746
t2* -0.1578447 -0.0006210818 0.007348956
```

```
summary(lm(mpg ~ horsepower, data = auto))
```

```
Call:
lm(formula = mpg ~ horsepower, data = auto)

Residuals:
     Min       1Q   Median       3Q      Max
-13.5710  -3.2592  -0.3435   2.7630  16.9240

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 39.935861   0.717499   55.66   <2e-16 ***
horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.906 on 390 degrees of freedom
Multiple R-squared:  0.6059,    Adjusted R-squared:  0.6049
F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

```
boot.fn <- function(data, index){
   coefficients(lm(mpg ~ horsepower + I(horsepower^2), data = data), subset = index)
}

set.seed(1)

boot(data = auto, statistic = boot.fn, R = 1000)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = auto, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
        original  bias    std. error
t1* 56.900099702       0           0
t2* -0.466189630       0           0
t3*  0.001230536       0           0
```

Conceptual

1.) Using the statistical properties of the variance as well as single variable calculus, derive 5.6. In other words, prove that α given by 5.6 does indeed minimize $Var(\alpha X + (1-\alpha)Y$

$Var(\alpha X + (1-\alpha)Y = \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY}$

2.) We now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of *n* observations.

a.) What is the probability that the first bootstrap observation is *not* the *j*th observation from the original sample?

$1 - \frac{1}{n}$

b.) What is the probability that the second bootstrap observation is *not* the *j*th observation from the original sample?

$1 - \frac{1}{n}$

c.) Argue that the probability that the *j*th observation is not in the bootstrap sample is $(1 - \frac{1}{n})^n$.

d.) When $n = 5$, what is the probability that the *j*th observation is in the bootstrap sample?

$1 - (1 - \frac{1}{5})^5 = .672$

e.) When $n = 100$, what is the probability that the *j*th observation is in the bootstrap sample?

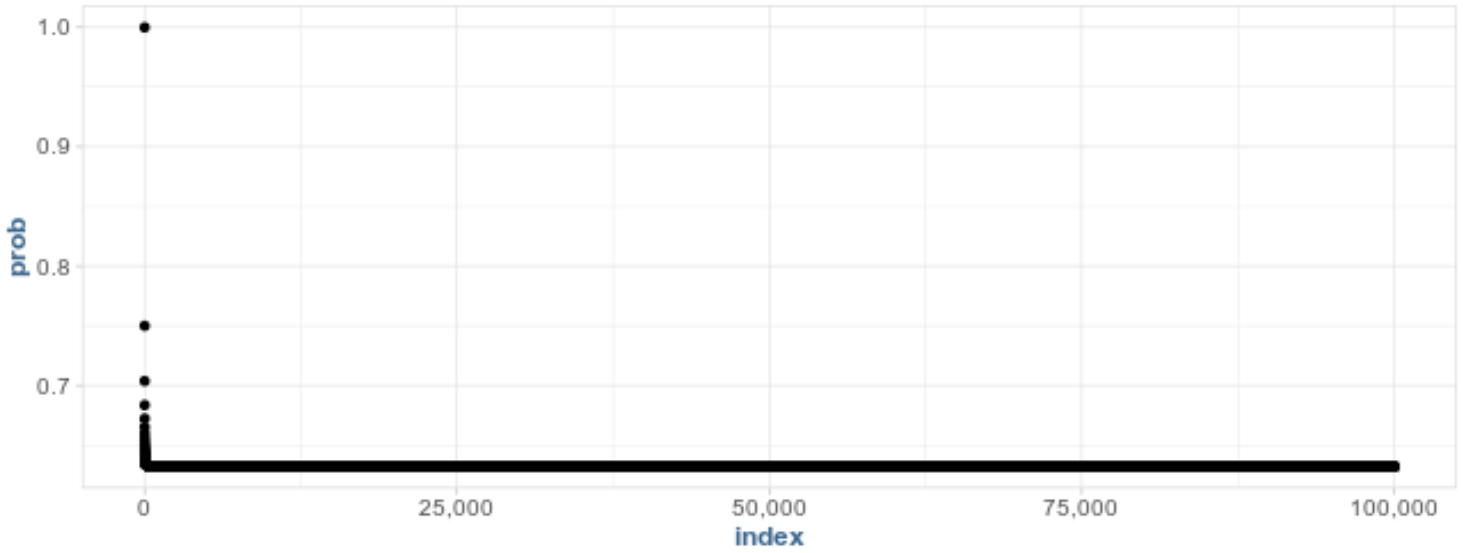
$1 - (1 - \frac{1}{100})^100 = .634$

f.) When $n = 1000$, what is the probability that the *j*th observation is in the bootstrap sample?

$1 - (1 - \frac{1}{1000})^{1000} = .632$

g.) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the *j*th observation is in the bootstrap sample. Comment on what you observe.

```
data <- data.table(index = 1:1e5)
data[, prob := 1 - (1 - 1/index)^index ]

ggplot(data, aes(index, prob)) +
   geom_point() +
   scale_x_continuous(labels = scales::comma)
```



Quickly approaches asymptote of .632

h.) We will now investigate the numerical probability that a bootstrap sample of size $n = 100$ contains the *j*th observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

```
store <- rep(NA, 1e5)

for(i in 1:1e5)
{
   store[i] <- sum(sample(1:100, rep = T) == 4) > 0
}
```

```
mean(store)
```

```
[1] 0.63329
```

3.) We now review k-fold cross-validation.

a.) Explain how k-fold cross validation is implemented:

The data is split up into k (roughly equal) parts, k-1 of which is used to train the model that is then tested on the hold-out set.

b.) What are the advantages and disadvantages of k-fold cross-validation relative to:

i.) The validation set approach?

The validation set approach has two main drawbacks compared to k-fold cross-validation. First, 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). Second, 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.

ii.) LOOCV?

k-Fold CV is computationally faster than LOOCV.

LOOCV cross-validation approach may give approximately unbiased estimates of the test error, since each training set contains n−1 observations; however, this approach has higher variance than k-fold cross-validation (since we are averaging the outputs of n fitted models trained on an almost identical set of observations, these outputs are highly correlated, and the mean of highly correlated quantities has higher variance than less correlated ones). So, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation; typically using k=5 or k=10 yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

4.) Suppose that we use some sttatistical learning method to make a prediction for the response Y for a particular value of the predictor X. Carefully describe how we might estimate the standard deviation of our prediction.

We may estimate the standard deviation of our prediction by using the bootstrap method. In this case, rather than obtaining new independant data sets from the population and fitting our model on those data sets, we instead obtain repeated random samples from the original data set. In this case, we perform sampling with replacement B times and then find the corresponding estimates and the standard deviation of those B estimates by using equation (5.8).

Applied

5.) In Chapter 4, we used logistic regression to predict the probability of default using income and balance on the default data set. We will now estimate the test error of this logistic regression model using the validation set approach.

```
set.seed(1)

default <- data.table(ISLR::Default)

default.split <- initial_split(default,
                               prop = .5,
                               strata = default)

default.train <- training(default.split)
default.test <- testing(default.split)
```

a.) Fit a logistic regression model that uses income and balance to predict default.

```
summary(fit1 <- glm(default ~ income + balance, family = binomial, data = default.train))
```

```
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = default.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3030  -0.1549  -0.0630  -0.0235   3.6608

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.074e+01  5.904e-01 -18.199   <2e-16 ***
income       8.921e-06  7.102e-06   1.256    0.209
balance      5.384e-03  3.135e-04  17.174   <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: 1429.88  on 4999  degrees of freedom
Residual deviance:  813.01  on 4997  degrees of freedom
AIC: 819.01
```

```
Number of Fisher Scoring iterations: 8
```

b.) Using the validation set approach, estimate the test error of this model.

```
default.test$pred <- ifelse( predict(fit1, newdata = default.test, type = "response") > .5, "Ye

with(default.test, mean(default != pred))
```

```
[1] 0.0266
```

c.) Repeat the process in b three times, using three different splits of the observations into a training set and a validation set.

```
set.seed(1)

trials <- 3

results <- numeric(trials)

for( i in 1:trials )
{
   default.split <- initial_split(default,
                                  prob = .5,
                                  strata = default)

   default.train <- training(default.split)
   default.test <- testing(default.split)

   model <- glm(default ~ income + balance, family = binomial, data = default.train)

   pred <- ifelse( predict(model, newdata = default.test, type = "response") > .5, "Yes", "No")

   results[i] <- mean(default.test$default != pred)
}

results
```

```
[1] 0.0288 0.0264 0.0208
```

The test error rates vary with sampling randomness.

d.) Now consider a logistic regression model that predicts the probability of default using income, balance and a dummy variable for student. Estimate the test error for this model using the validation set approach.

```
set.seed(1)

default <- data.table(ISLR::Default)

# one-hot encode student
default[, ':='(student_Yes = ifelse(student == "Yes", 1, 0), student_No = ifelse(student == "No
```

```
default.split <- initial_split(default,
                               prob = .5,
                               strata = default)

default.train <- training(default.split)
default.test <- testing(default.split)

summary(fit2 <- glm(default ~ income + balance + student_Yes, data = default.train, family = bi
```

```
Call:
glm(formula = default ~ income + balance + student_Yes, family = binomial,
    data = default.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3645  -0.1427  -0.0558  -0.0204   3.6757

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.053e+01  5.721e-01 -18.405   <2e-16 ***
income      -5.858e-06  9.695e-06  -0.604   0.5457
balance      5.669e-03  2.712e-04  20.907   <2e-16 ***
student_Yes -6.070e-01  2.702e-01  -2.247   0.0246 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2138.0  on 7499  degrees of freedom
Residual deviance: 1170.9  on 7496  degrees of freedom
AIC: 1178.9

Number of Fisher Scoring iterations: 8
```

```
default.test$pred <- ifelse(predict(fit2, newdata = default.test, type = "response") > .5, "Yes

with(default.test, mean(default != pred))
```

```
[1] 0.0276
```

Student dummy coding slighly reduces the test error rate for the probability of default model.

6.) We continue to consider the use of a logistic regression model to predict the probability of "default" using "income" and "balance" on the "Default" data set. In particular, we will now computes estimates for the standard errors of the "income" and "balance" logistic regression coefficients in two different ways : (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

a.) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with "income" and "balance" in a multiple logistic regression model that uses both predictors.

```
set.seed(1)

summary(fit1 <- glm(default ~ income + balance, data = default.train, family = binomial))
```

```
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = default.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3697  -0.1440  -0.0575  -0.0213   3.7176

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.117e+01  5.035e-01 -22.182   <2e-16 ***
income       1.139e-05  5.856e-06   1.945   0.0518 .
balance      5.580e-03  2.656e-04  21.010   <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: 2138.0  on 7499  degrees of freedom
Residual deviance: 1175.9  on 7497  degrees of freedom
AIC: 1181.9

Number of Fisher Scoring iterations: 8
```

b.) Write a function, boot.fn(), that takes as input the default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
boot.fn <- function(data, index) {
   coefficients(glm(default ~ income + balance, family = binomial, data = data, subset = index)
}
```

c.) Use boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficents for income and balance.

```
boot(data = default, statistic = boot.fn, R = 1000)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = default, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01 -3.945460e-02 4.344722e-01
t2*  2.080898e-05  1.680317e-07 4.866284e-06
t3*  5.647103e-03  1.855765e-05 2.298949e-04
```

7.) We saw that cv.glm() function can be used in order to compute the LOOCV test error estimate. Alternatively, one could compute those quantities using just the glm() and predict.glm() functions, and a for loop. You will now take this approach in order to compute the LOOCY error for a simple logistic regresion model on the Weekly data set.

```
set.seed(1)

weekly <- data.table(ISLR::Weekly)

weekly.split <- initial_split(weekly, prop = .5)

weekly.train <- training(weekly.split); weekly.test <- testing(weekly.split)
```

a.) Fit a logistic regression model that predicts Direction using the Lag1 and Lag2.

```
summary(fit1 <- glm(Direction ~ Lag1 + Lag2, data = weekly.train, family = binomial))
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = weekly.train)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.466  -1.315   1.004   1.037   1.219

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.34748    0.08788   3.954 7.69e-05 ***
Lag1        -0.03219    0.03805  -0.846    0.398
```

```
Lag2         0.01754    0.03906   0.449    0.653
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 739.58  on 544  degrees of freedom
Residual deviance: 738.48  on 542  degrees of freedom
AIC: 744.48

Number of Fisher Scoring iterations: 4
```

b.) Fit a logistic regresion model that predicts direction using Lag1 and Lag2 using *all but the first observation.*

```
holdout <- weekly[1]

summary(fit1 <- glm(Direction ~ Lag1 + Lag2, data = weekly[-1], family = binomial))
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = weekly[-1])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6258  -1.2617   0.9999   1.0819   1.5071

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.22324    0.06150   3.630 0.000283 ***
Lag1        -0.03843    0.02622  -1.466 0.142683
Lag2         0.06085    0.02656   2.291 0.021971 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1494.6  on 1087  degrees of freedom
Residual deviance: 1486.5  on 1085  degrees of freedom
AIC: 1492.5

Number of Fisher Scoring iterations: 4
```

c.) Use the model from the previous to predict the first observation.

```
holdout$Direction == ifelse( predict(fit1, newdata = holdout, type = "response") > .5, "Up", "I
```

```
[1] FALSE
```

The model did not predict this correctly.

d.) Write a for loop from i = 1 to i = n, where n is the number of observations in the data set, that performs each of the following steps:

i.) Fit a logistic regresion model using all but the *i*th observation. ii.) Compute the posterior probability of the market moving up for the ith observation.

```
n <- nrow(weekly)
results <- logical()
for(i in 1:n)
{
    model <- glm(Direction ~ Lag1 + Lag2, data = weekly[-i,], family = binomial)
    pred <- ifelse( predict(model, newdata = weekly[i,], type = "response") > .5, "Up", "Down")
    results[i] <- weekly[i,]$Direction != pred
}
```

iii.) Take the average of the n numbers obtained in d.) in order to obtain the LOOCV estimate for the test error:

```
mean(results)
```

```
[1] 0.4499541
```

8.)

We will now perform cross-validation on a simulated data set.

a.) Generate a simulated data set as follows:

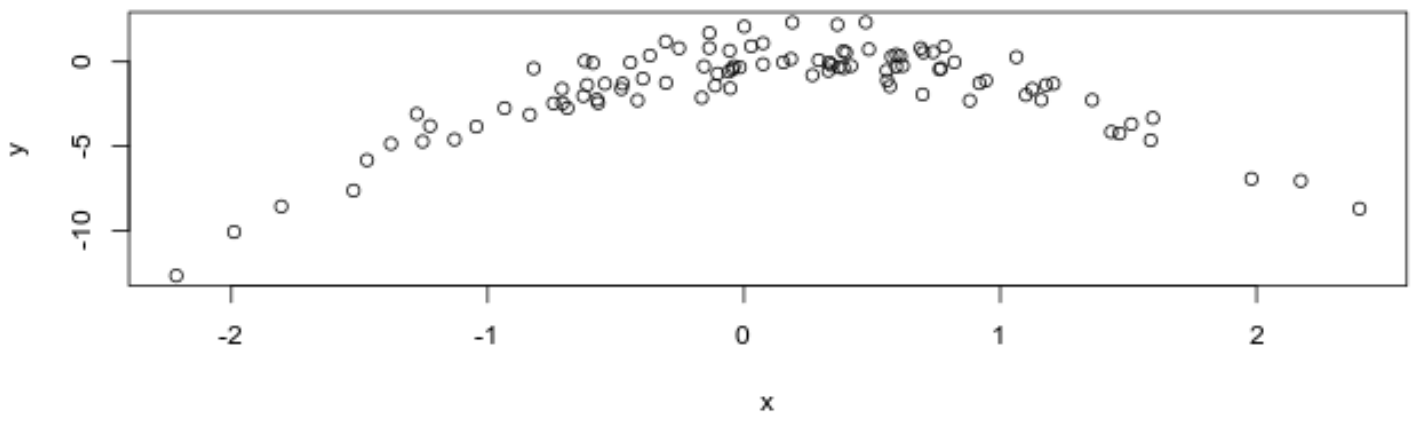
```
set.seed(1)
x <- rnorm(100); y <- x - 2 * x^2 + rnorm(100)

data <- data.table(x, y)
```

In this data set, what is n and what is p?

n = 100, p = 2, $Y = X - 2 * X^2 + \epsilon$

b.) Create a scatterplot of x, y

```
plot(x, y)
```



c.) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using the least squares:

i.) $Y = \beta_0 + \beta_1 X + \epsilon$

```
set.seed(1)
fit1 <- glm(y ~ x, data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 7.288162
```

ii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

```
set.seed(1)
fit1 <- glm(y ~ poly(x, 2), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9374236
```

iii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

```
set.seed(1)
fit1 <- glm(y ~ poly(x, 3), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9566218
```

iiii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 \epsilon$

```
set.seed(1)
fit1 <- glm(y ~ poly(x, 4), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9539049
```

d.) Repeat c using another random seed, and report results.

i.) $Y = \beta_0 + \beta_1 X + \epsilon$

```
set.seed(100)
fit1 <- glm(y ~ x, data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 7.288162
```

ii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

```
set.seed(100)
fit1 <- glm(y ~ poly(x, 2), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9374236
```

iii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

```
set.seed(100)
fit1 <- glm(y ~ poly(x, 3), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9566218
```

iiii.) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 \epsilon$

```
set.seed(100)
fit1 <- glm(y ~ poly(x, 4), data = data)
cv.glm(data, fit1)$delta[1]
```

```
[1] 0.9539049
```

Re