Chapter 5 Excercises

Jonathan Bryan

May 19, 2018

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

$$\underset{\alpha}{\operatorname{argmin}}\, Var(\alpha X + (1-\alpha)Y)$$

$$\begin{aligned}
Var(\alpha X + (1-\alpha)Y) &= \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y)) \\
Var(\alpha X + (1-\alpha)Y) &= \alpha^2\sigma_X^2 + (1-2\alpha+\alpha^2)\sigma_Y^2 + (2\alpha - 2\alpha^2)\sigma_{XY}^2 \\
\frac{d}{d\alpha}Var(\alpha X + (1-\alpha)Y) &= 2\alpha\sigma_X^2 + (-2+2\alpha)\sigma_Y^2 + (2-4\alpha)\sigma_{XY}^2 \\
\frac{d}{d\alpha}Var(\alpha X + (1-\alpha)Y) &= 2\alpha\sigma_X^2 - 2\sigma_Y^2 + 2\alpha\sigma_Y^2 + 2\sigma_{XY}^2 - 4\alpha\sigma_{XY}^2 \\
2\alpha\sigma_X^2 - 2\sigma_Y^2 + 2\alpha\sigma_Y^2 + 2\sigma_{XY}^2 - 4\alpha\sigma_{XY}^2 &= 0 \\
2\alpha\sigma_X^2 + 2\alpha\sigma_Y^2 - 4\alpha\sigma_{XY}^2 &= 2\sigma_Y^2 - 2\sigma_{XY}^2 \\
2\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}^2) &= 2\sigma_Y^2 - 2\sigma_{XY}^2 \\
\alpha &= \frac{\sigma_Y^2 - \sigma_{XY}^2}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}^2}
\end{aligned}$$

2. We will 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 jth observation from the original sample? Justify your answer.

$$Pr(B^1 \neq j) = \frac{n-1}{n}$$

(b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

$$Pr(B^2 \neq j) = \frac{n-1}{n}$$

(c) Argue that the probability that the jth observation is not in the bootstrap sample is $(1-1/n)/^n$.

$$Pr(B^1 \neq j, B^2 \neq j, ..., B^n \neq j) = \prod_{i=1}^{n}\frac{n-1}{n} = \left(\frac{n-1}{n}\right)^n$$

(d) When n = 5, what is the probability that the jth observation is in the bootstrap sample?

$$Pr(j \in B|n = 5) = 1 - \binom{5}{0}(1/5)^0(4/5)^5 = 1 - .328 = .672$$

(e) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

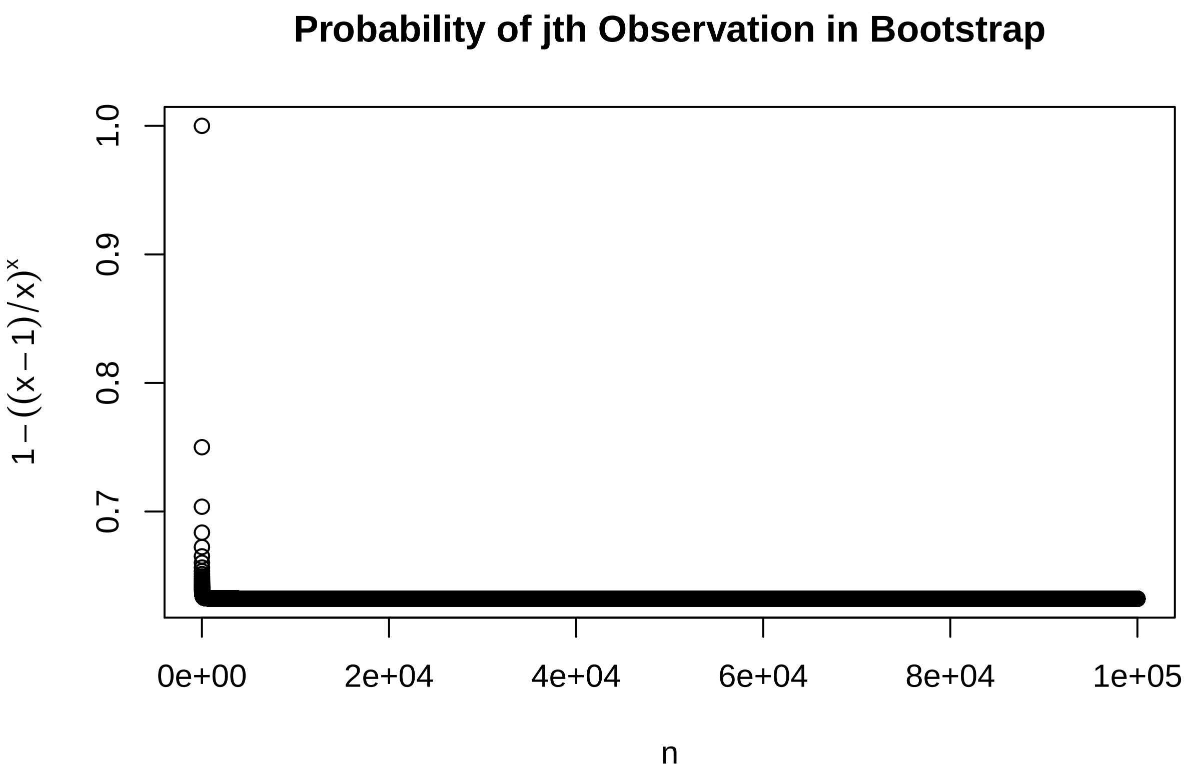
$$Pr(j \in B|n = 100) = 1 - (99/100)^{100} = 1 - .366 = .634$$

(e) When n = 10, 000, what is the probability that the jth observation is in the bootstrap sample?

$$Pr(j \in B|n = 10000) = 1 - (9999/10000)^{10000} = 1 - .367 = .632$$

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

```
x = seq(1,100000)
y = 1 - ((x - 1)/x)^x
plot(x,y, main = "Probability of jth Observation in Bootstrap",
          xlab = "n",
          ylab = expression(1 - ((x - 1)/x)^x))
```



The probability that the jth observation is in the bootstrap approaches $\approx \frac{2}{3}$ as n increaes.

(h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth 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, 10000)
for(i in 1:10000){
  store[i] = sum(sample(1:100, rep=TRUE) ==4) > 0
}
mean(store)
```

```
## [1] 0.6357
```

On average the probability that the jth observation is in the bootstrap sample is .64,

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

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

A integer k is selected and a $\frac{k-1}{k}$ proportion of the data is randomly selected as the training set while the $\frac{1}{k}$ rest of the data serves as the holdout set. The model is trained on the training set and validated using the holdout set k times. An estimate of the test error is obtained by averaging the total error (e.g. RMSE, classification error) of each model.

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

i. The validation set approach?

k-fold cross-validation will typically result in a better estimate of the test error rate (especially if k is between 5 or 10) by reducing the bias and variance of the estimate as more data is used to train the model. However, k-fold cross-validation requires possibly fine tuning the optimal k to derive the best test error estimated, while the validation approach always divides the dataset into two parts and is thus computationally faster.

ii. LOOCV?

While LOOCV has the lowest bias of the cross-validation estimators of test error rates, the LOOCV test error estimate will vary more with different datasets of size n from the same population due to using nearly the same training data for each of the n models and thus having outputs that are highly correlated. k-fold cross-validation is usually preferred as it has a better balance of bias and variance performance. LOOCV is also computationally more expensive when $k < n$.

4. Suppose that we use some statistical 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 could repeat the process of fitting the statistical model by drawing a random observation and associated response from the data (X^*, Y^*) n number of times. Each fitted model, could then use the particular value of X^p to to predict $\hat{Y^p}$. We would collect n prediction estimates using X^p and calculated the standard deviation using $\sqrt{\{\frac{1}{n-1}\Sigma_{i=1}^{n}(\hat{Y^p} - \bar{\hat{Y^p}})}$.

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. Do not forget to set a random seed before beginning your analysis.

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

```r
library(ISLR)
library(stargazer)
```

```
## Warning: package 'stargazer' was built under R version 3.4.3
```

```r
set.seed(1)
lr = glm(as.numeric(default)-1 ~ income + balance, family = "binomial", data = Default)
stargazer(lr, header=FALSE)
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

```r
set.seed(1)
train = sample(seq(1,nrow(Default),by=1),size = nrow(Default)/2 , replace=FALSE)
default.train = Default[train,]
default.validation = Default[-train,]
```

Table 1:

	Dependent variable:
	as.numeric(default) - 1
income	0.00002***
	(0.00000)
balance	0.006***
	(0.0002)
Constant	−11.540***
	(0.435)
Observations	10,000
Log Likelihood	−789.483
Akaike Inf. Crit.	1,584.966
Note:	*p<0.1; **p<0.05; ***p<0.01

ii. Fit a multiple logistic regression model using only the training observations.

```
lr2 = glm(as.numeric(default) - 1 ~ income + balance, family = "binomial", data = default.train)
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

```
lr2.pred = predict(lr2, newdata = default.validation, type = "response")
lr2.pred[lr2.pred > 0.5] = 1
lr2.pred[lr2.pred <= 0.5] = 0
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
sum(lr2.pred != as.numeric(default.validation$default)-1)/nrow(default.validation)
```

```
## [1] 0.0286
```

The validation set error rate is 2.86%.

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

```
set.seed(2)
class.error = rep(NA,3)
for ( i in 1:3){
  train = sample( seq(1, nrow(Default), by = 1), size = nrow(Default)/2, replace = FALSE)
  default.train = Default[train,]
  default.validation = Default[-train,]
  lr3 = glm(as.numeric(default) - 1 ~ income + balance, family = "binomial",
            data = default.train)
  lr3.pred = predict(lr3, newdata = default.validation, type = "response")
  lr3.pred[lr3.pred > 0.5] = 1
  lr3.pred[lr3.pred <= 0.5] = 0
  class.error[i] = sum(lr3.pred != as.numeric(default.validation$default) - 1)/nrow(default.validation)
}
range(class.error)
```

```
## [1] 0.0258 0.0278
```

```
mean(class.error)
```

```
## [1] 0.02706667
```

The validation set misclassification rate ranged from 2.58% to 2.78% and the mean was 2.71%.

(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. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

```
set.seed(3)
train = sample( seq(1, nrow(Default), by = 1),
                size = nrow(Default)/2,
                replace = FALSE)
default.train = Default[train,]
default.validation = Default[-train,]
lr4 = glm(as.numeric(default) - 1 ~ income + balance + as.factor(student),
          family = "binomial",
          data = default.train)
lr4.pred = predict(lr4,
                   newdata = default.validation,
                   type = "response")
lr4.pred[lr4.pred > 0.5] = 1
lr4.pred[lr4.pred <= 0.5] = 0
sum(lr4.pred != as.numeric(default.validation$default) - 1)/nrow(default.validation)
```

```
## [1] 0.0248
```

Using a student dummy variable lowered the validation set misclassification rate to 2.48%.

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 compute 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.

```
lr5 = glm( as.numeric(default) - 1 ~ income + balance,
           family = "binomial",
           data = Default)
stargazer(lr5, header=FALSE)
```

Table 2:

	Dependent variable:
	as.numeric(default) - 1
income	0.00002***
	(0.00000)
balance	0.006***
	(0.0002)
Constant	−11.540***
	(0.435)
Observations	10,000
Log Likelihood	−789.483
Akaike Inf. Crit.	1,584.966
Note:	*p<0.1; **p<0.05; ***p<0.01

(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){
  lr.boot = glm( as.numeric(default) - 1 ~ income + balance,
                 family = "binomial",
                 data = Default,
                 subset = index)
  coef.est = summary(lr.boot)$coefficients[2:3,1]
  coef.est
}
```

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

```
library(boot)
set.seed(1)
boot.se = boot(data = Default, statistic = boot.fn, R = 100)
boot.se
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 100)
##
##
## Bootstrap Statistics :
##         original        bias    std. error
## t1* 2.080898e-05  6.715005e-08 4.12774e-06
## t2* 5.647103e-03 -5.733883e-05 2.10566e-04
```

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

When using the glm() function, the estimated standard errors for income and balance are 4.99e-06 and 2.26e-04 respectively. The bootstrap estimated standard errors for income and balance are 4.13e-06 and 2.11e-04 respectively. Both of the boostrap estimates are lower than the estimates from part (a), suggesting a deviation from the hypothesized linear relationship between the predictors and the responses, as the estimated variance using the glm() is likely overinflated.

7. In Sections 5.3.2 and 5.3.3, we saw that the 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 LOOCV error for a simple logistic regression model on the Weekly data set. Recall that in the context of classification problems, the LOOCV error is given in (5.4).

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

```r
lr6 = glm(as.numeric(Direction) - 1 ~ Lag1 + Lag2, family = "binomial", data = Weekly)
```

(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.

```r
lr7 = glm(as.numeric(Direction) - 1 ~ Lag1 + Lag2, family = "binomial", data = Weekly[-1,])
```

(c) Use the model from (b) to predict the direction of the first observation. You can do this by predicting that the first observation will go up if $P(Direction = "Up"|Lag1, Lag2) > 0.5$. Was this observation correctly classified?

```r
lr7.pred = predict(lr7, newdata = Weekly[1,], type = "response")
lr7.pred[lr7.pred > 0.5] = 1
lr7.pred[lr7.pred <= 0] = 0
lr7.pred == as.numeric(Weekly[1,"Direction"]) - 1
```

```
##     1
## FALSE
```

The observation's predicted response was incorrectly classified as default.

(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 regression model using all but the ith observation to predict Direction using Lag1 and Lag2.

ii. Compute the posterior probability of the market moving up for the ith observation.

iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.

iv. Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

```r
error = rep(NA,nrow(Weekly))
test = rep(NA,nrow(Weekly))
for (i in 1:nrow(Weekly)){
  lr.boot = glm(as.numeric(Direction) - 1 ~ Lag1 + Lag2, family="binomial", data = Weekly[-i,])
  lr.boot.pred = predict(lr.boot, newdata = Weekly[i,], type = "response")
  lr.boot.pred[lr.boot.pred > 0.5] = 1
  lr.boot.pred[lr.boot.pred <= 0.5] = 0
  error[i] = lr.boot.pred == as.numeric(Weekly[i,"Direction"])-1
}
```

(e) Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

```r
1 - mean(error)
```

```
## [1] 0.4499541
```

The LOOCV estimate for the test error is approximately 45%.

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

(a) Generate a simulated data set as follows:

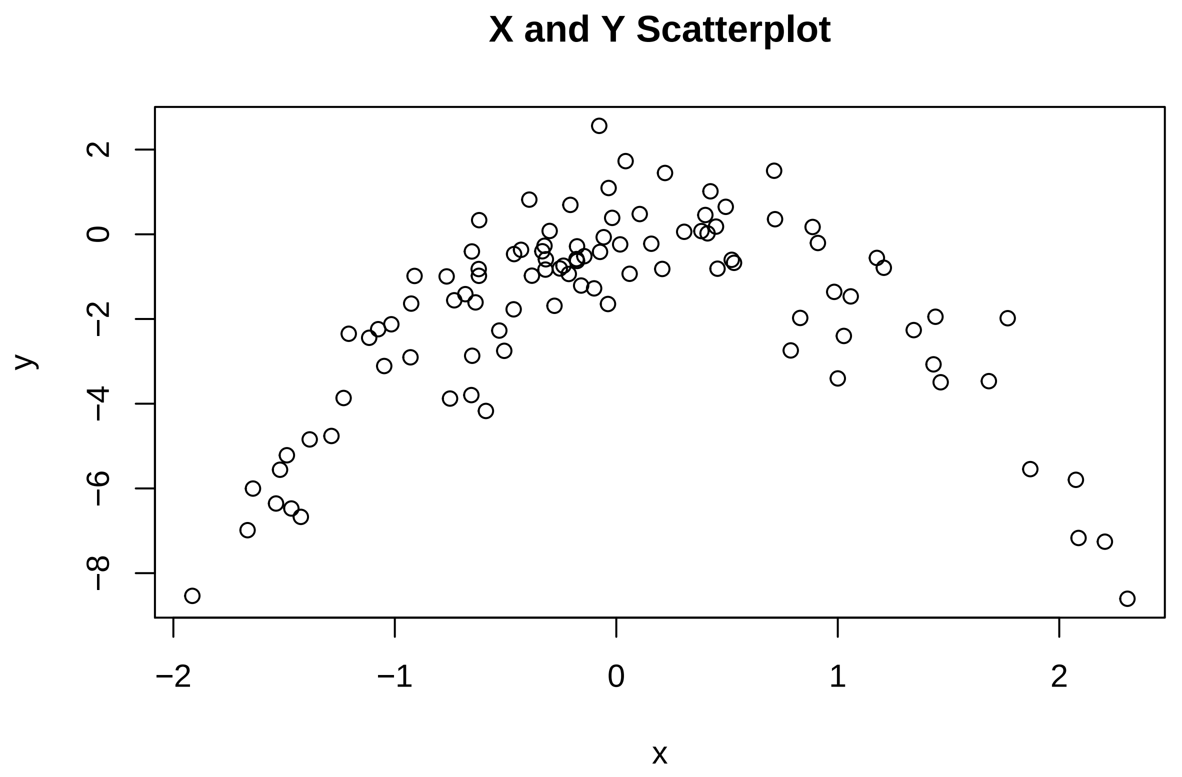
```r
set.seed (1)
y=rnorm(100)
x=rnorm(100)
y=x-2* x^2+ rnorm (100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

In this model there are 100 simulated observations and two predictors, therefore, $n = 100$ and $p = 2$. The model is $Y = X - 2X^2 + \epsilon$, $\epsilon \sim N(0, 1)$.

(b) Create a scatterplot of X against Y . Comment on what you find.

```r
plot(x,y, main = "X and Y Scatterplot")
```



There is a quadratic relationship with X and Y.

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

```r
set.seed(1)
sim_df = data.frame(y = y, x=x)
i = cv.glm(glmfit = glm(y ~ x), data = sim_df)$delta[1]
ii = cv.glm(glmfit = glm(y ~ poly(x,2)), data = sim_df)$delta[1]
iii = cv.glm(glmfit = glm(y ~ poly(x,3)), data = sim_df)$delta[1]
iv = cv.glm(glmfit = glm(y ~ poly(x,4)), data = sim_df)$delta[1]

table1 = round(data.frame(i = i, ii = ii, iii = iii, iv= iv),3)
knitr::kable(table1, caption = "Comparison of Models for LOOCV Seed(1)")
```

Table 3: Comparison of Models for LOOCV Seed(1)

i	ii	iii	iv
5.891	1.087	1.103	1.115

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```r
set.seed(2)
i = cv.glm(glmfit = glm(y ~ x), data = sim_df)$delta[1]
ii = cv.glm(glmfit = glm(y ~ poly(x,2)), data = sim_df)$delta[1]
iii = cv.glm(glmfit = glm(y ~ poly(x,3)), data = sim_df)$delta[1]
iv = cv.glm(glmfit = glm(y ~ poly(x,4)), data = sim_df)$delta[1]

table2 = round(data.frame(i = i, ii = ii, iii = iii, iv= iv),3)
knitr::kable(table2, caption = "Comparison of Models for LOOCV Seed(2)")
```

Table 4: Comparison of Models for LOOCV Seed(2)

i	ii	iii	iv
5.891	1.087	1.103	1.115

The results are the same for parts (c) and (d). This is because LOOCV evaluates the same $n-1$ data to predict the holdout sample, even if the order of the selected holdout point changes with each random seed. The overall LOOCV will always be the same for a given model unless the underlying data set changes.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The quadratic model has the lowest LOOCV error. This is expected given the quadratic relationship between the simulated Y and X as seen in part (b).

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```r
stargazer(glm(y ~ x), header=FALSE)
```

Table 5:

	Dependent variable:
	y
x	0.243
	(0.248)
Constant	−1.819***
	(0.236)
Observations	100
Log Likelihood	−227.843
Akaike Inf. Crit.	459.687
Note:	*p<0.1; **p<0.05; ***p<0.01

```r
stargazer(glm(y ~ poly(x,2)), header=FALSE)
```

Table 6:

	Dependent variable:
	y
poly(x, 2)1	2.316**
	(1.032)
poly(x, 2)2	$-$21.059***
	(1.032)
Constant	$-$1.828***
	(0.103)
Observations	100
Log Likelihood	$-$144.555
Akaike Inf. Crit.	295.110
Note:	*p<0.1; **p<0.05; ***p<0.01

```r
stargazer(glm(y ~ poly(x,3)), header=FALSE)
```

Table 7:

	Dependent variable:
	y
poly(x, 3)1	2.316**
	(1.037)
poly(x, 3)2	$-$21.059***
	(1.037)
poly(x, 3)3	$-$0.305
	(1.037)
Constant	$-$1.828***
	(0.104)
Observations	100
Log Likelihood	$-$144.510
Akaike Inf. Crit.	297.020
Note:	*p<0.1; **p<0.05; ***p<0.01

```r
stargazer(glm(y ~ poly(x,4)), header=FALSE)
```

We observe that the quadratic coefficients are significant, while the other terms are not. This consistent with the LOOCV eror results, where the best model was the quadratic model.

Table 8:

	Dependent variable:
	y
poly(x, 4)1	2.316**
	(1.041)
poly(x, 4)2	$-$21.059***
	(1.041)
poly(x, 4)3	$-$0.305
	(1.041)
poly(x, 4)4	$-$0.493
	(1.041)
Constant	$-$1.828***
	(0.104)
Observations	100
Log Likelihood	$-$144.392
Akaike Inf. Crit.	298.784
Note:	*p<0.1; **p<0.05; ***p<0.01

9. We will now consider the Boston housing data set, from the MASS library.

(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate $\hat{\mu}$.

```
library(MASS)
mean(Boston$medv)
```

```
## [1] 22.53281
```

$\hat{\mu} = 22.533$.

(b) Provide an estimate of the standard error of $\hat{\mu}$. Interpret this result.

```
sqrt(var(Boston$medv))/(sqrt(nrow(Boston)))
```

```
## [1] 0.4088611
```

$\hat{\sigma\mu} = 0.408$. The sample mean has an estimated standard deviation of 0.408.

(c) Now estimate the standard error of $\hat{\mu}$ using the bootstrap. How does this compare to your answer from (b)?

```
set.seed(1)
boot.mean = function(data,index){
  return (mean(data[index]))
```

```r
}
boot(data = Boston$medv, statistic = boot.mean, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.mean, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.008517589   0.4119374
```

The bootstrap estimate for is $\hat{\sigma \mu} = 0.412$. This is very close to the estimate in part (b) but slightly larger.

(d) Based on your bootstrap estimate from (c), provide a 95% confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).

```r
c(22.53281 - 2*(0.4119374),22.53281 + 2*(0.4119374) )
```

```
## [1] 21.70894 23.35668
```

```r
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

Bootstrap 95% confidence interval (22.71,23.37) is larger than the t-test (21.73 23.34) derived confiidence interval.

(e) Based on this data set, provide an estimate, $\hat{\mu_{med}}$, for the median value of medv in the population.

```r
median(Boston$medv)
```

```
## [1] 21.2
```

(f) We now would like to estimate the standard error of $\hat{\mu_{med}}$. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.

```
boot.med = function(data,index){
  return(median(data[index]))
}
boot(data = Boston$medv, statistic = boot.med, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.med, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0098   0.3874004
```

The bootstrap estimate for the median and the standard error of the median is 21.2 and 0.374 respectively. Both the estimate of the median and its standard error are smaller than the mean estimates.

(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity $\hat{\mu_{0.1}}$. (You can use the quantile() function.)

```
quantile(Boston$medv, probs = 0.1)
```

```
##   10%
## 12.75
```

(h) Use the bootstrap to estimate the standard error of ˆ??0.1. Comment on your findings.

```
boot.0.1 = function(data,index){
  return(quantile(data[index], probs = 0.1 ))
}
boot(data = Boston$medv, statistic = boot.0.1, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.0.1, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00515   0.5113487
```

The bootstrap estimate and standard error for the 10th percentile value of medv is 12.75 and 0.49, respectively.