Homework 2

Lydia Strebe

February 19, 2019

Problem 1

1. Since the data set has n points in it, the probability that the jth data point from the original data set *is* the first entry in the bootstrapped sample is $1/n$. Therefore, the probability that the jth data point from the original data set is *not* the first entry is $1 - 1/n$.
2. The probability that the jth data point is not the *second* entry is also $1 - 1/n$.
3. The two probabilities are independent because the sample is taken with replacement. Because the probabilities are independent, we can multiply them to get the probablity that the jth data point is not the first or second entry: $(1 - 1/n)^2$.
4. The probability that the jth data point is not in the bootstrapped sample is $(1 - 1/n)^n$.
5. To show that $\lim_{n\to\infty}(1 - 1/n)^n = 1/e$, we start with

$$(1 - 1/n)^n = e^{ln(1-1/n)^n}$$

$$ln(1 - 1/n)^n = n * ln(1 - 1/n) = ln(1 - 1/n)/(1/n)$$

$$\lim_{n\to\infty} ln(1 - 1/n)/(1/n)$$

Here we use L'Hopital's rule by taking the derivative of the numerator and denominator. By canceling we get:

$$\lim_{n\to\infty} -1/(1 - 1/n) = -1$$

Therefore:

$$\lim_{n\to\infty}(1 - 1/n)^n = \lim_{n\to\infty} e^{ln(1-1/n)^n} = e^{-1} = 1/e$$

Problem 2

1.

$$E[pd - cd] = E[(p - c)(\alpha e^{\gamma p} + \sigma\epsilon)]$$

E[p{\alpha}e^{{\gamma}p]+E[p{\sigma}\epsilon]-E[c{\alpha}e^{{\gamma}p]-E[c{\sigma}\epsilon

Since ϵ is a Gaussian random variable with a mean of 0. We know that $E[p\sigma\epsilon] = 0$ and $E[c\sigma\epsilon] = 0$. This leaves

$$(p - c)(\alpha e^{\gamma p})$$

To find the optimal price, we take the derivative with respect to p, set it equal to 0, and solve for p:

$$\alpha e^{\gamma p} - \alpha\gamma(p - c)e^{-\gamma p} = 0$$

$$\alpha e^{\gamma p} = \gamma(p - c)\alpha e^{\gamma p}$$

$$1 = \gamma(p - c)$$

$$p* = (1 + \gamma c)/\gamma = 1/\gamma + c$$

2. The following code simulates data for price and demand

```
alpha=100
gamma=3
sigma=3/2
c=2/3
pstar=(1/gamma)+c
price_data = runif(100)*0.5*pstar + 0.75*pstar
demand_data = alpha*exp(-gamma*price_data)+sigma*rnorm(100)
```

3. The following code uses 10-fold cross validation to find the best model for our price and demand data. We will compare a linear relationship between price and demand to a quadratic relationship between price and demand.

```
require(boot)
```

```
## Loading required package: boot
```

```
#Put data into dataframe
retailer = data.frame("Demand"=demand_data, "Price"=price_data)

#Linear model
lmodel = glm(Demand~Price,data=retailer)

#Quadratic model
qmodel = glm(Demand~poly(Price,2,raw=T),data=retailer)

lin_cv10 = cv.glm(retailer,lmodel,K=10)$delta[1]
quad_cv10 = cv.glm(retailer,qmodel,K=10)$delta[1]

lin_cv10
```

```
## [1] 2.087553
```

```
quad_cv10
```

```
## [1] 2.057435
```

We see that the mean squared error is smaller for the quadratic model.

4. We can use this model to develop for the `approximate' optimal selling price.

We start with $demand = \hat{\beta}_0 + \hat{\beta}_1 * p + \hat{\beta}_2 * p^2$

$$p * demand - c * demand = \hat{\beta}_0 * p + \hat{\beta}_1 * p^2 + \hat{\beta}_2 * p^3 - c * \hat{\beta}_0 - c * \hat{\beta}_1 * p - c * \hat{\beta}_2 * p^2$$

We take the derivative and set it equal to 0:

$$(3 * \hat{\beta}_2) * p^2 + (2 * \hat{\beta}_1 - 2 * c * \hat{\beta}_2) * p + (\hat{\beta}_0 - c * \hat{\beta}_1) = 0$$

We then solve for p using the quadratic equation:

$$p = (-(2\hat{\beta}_1 - 2c * \hat{\beta}_2) \pm \sqrt{(2\hat{\beta}_1 - 2c * \hat{\beta}_2)^2 - 4 * (3\hat{\beta}_2) * (\hat{\beta}_0 - c * \hat{\beta}_1)})/(2 * 3\hat{\beta}_2)$$

This may give us two values for p, so we take the second derivative to find the maximum:

$$2\hat{\beta}_1 + 6\hat{\beta}_2 p - 2c * \hat{\beta}_2 < 0$$

5. Below is a function to calculate the optimal price using the formula above:

```
#Better optimal price function
opt_price_fun = function(data,index){

  #run regression
  model = glm(Demand~poly(Price,2,raw=T),data=data, subset=index)

  #get regression coefficients
  b0=summary(model)$coef[1]
  b1=summary(model)$coef[2]
  b2=summary(model)$coef[3]

  #prep for quadratice equation
  A=3*b2
  B=(2*b1-2*c*b2)
  C=(b0-c*b1)

  #calculate discriminant
  d=(B^2-4*A*C)

  #quadratic with 2 real roots (discriminant>0)
  if(d > 0){
    p1 = (-B+sqrt(d))/(2*A)
    p2 = (-B-sqrt(d))/(2*A)

    #check second derivative
    if(2*b1+6*b2*p1-2*b2*c < 0){
      max(0,p1)
    }
    else{
      max(0,p2)
    }
  }

  #quadratic with 1 root (discriminant=0)
  else if(d == 0){
    p = -B/(2*A)
    max(0,p)
  }

  #no real roots (discriminant<0)
  else {
    p=0
    p
  }
}

boot_price = boot(retailer,opt_price_fun,R=1000)
boot_price
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = retailer, statistic = opt_price_fun, R = 1000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.9630955 -0.0002261918  0.01816591
```

```
quantile(boot_price$t[,1], probs=c(.025,.975))
```

```
##      2.5%     97.5%
## 0.9260605 0.9962664
```

As you can see, I created 1000 bootstrapped data sets by sampling the data I simulated in step 2 and calculated a 95% confidence interval for this price.