Stat 208 - Linear Models - HW 1

Jordan Berninger

4/8/2020

Question 1

We simply read in the data and fit a linear model using $lm()$ with an intercept term. We see the fitted parameters in the estimate column of the model summary. We also compute the vector $\hat{\beta}$ with the formula from class and confirm that these estimated parameters match those returned by $lm()$.

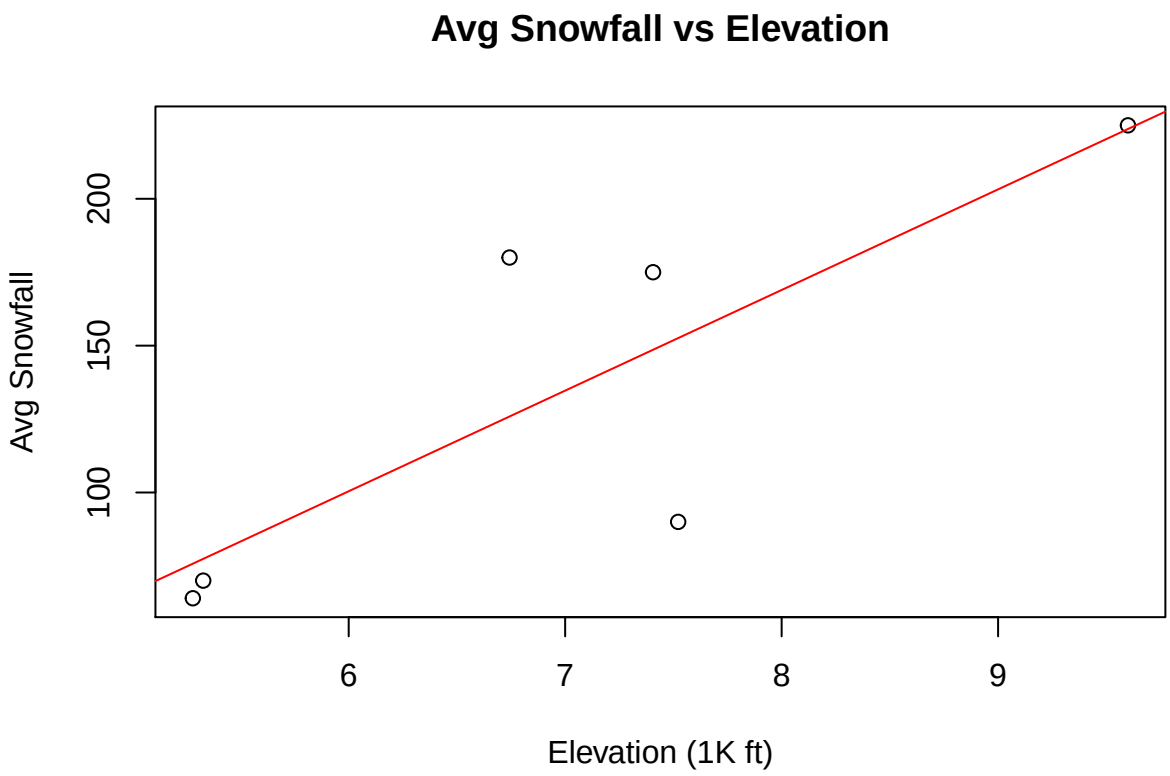
```
snow <- data.frame(city = c("Denver", "Boulder", "Estes Park",
                            "Breckenridge", "Steamboat Springs", "Aspen"),
                   elevation = c(5.280, 5.328, 7.522, 9.600, 6.743, 7.406),
                   avg_snow = c(64, 70, 90, 225, 180, 175))

m <- lm(data = snow, avg_snow ~ elevation)
summary(m)
```

```
##
## Call:
## lm(formula = avg_snow ~ elevation, data = snow)
##
## Residuals:
##       1       2       3       4       5       6
## -11.767  -7.412 -62.574   1.239  54.113  26.400
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -105.11      86.96  -1.209   0.2933
## elevation      34.26      12.19   2.810   0.0483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.98 on 4 degrees of freedom
## Multiple R-squared:  0.6638, Adjusted R-squared:  0.5798
## F-statistic: 7.898 on 1 and 4 DF,  p-value: 0.0483
```

We note that the model summary shows a relatively high p-value for the intercept parameter (μ in the model notation), indicating a linear model through the origin may be more appropriate. We also plot the data and our fitted linear model and note this model fits the extreme data points well, the middle data points, not so well.

```
plot(x = snow$elevation, y = snow$avg_snow, main = "Avg Snowfall vs Elevation",
     xlab = "Elevation (1K ft)", ylab = "Avg Snowfall")
abline(a = m$coefficients[1], b = m$coefficients[2], col = "red")
```



To get the model coefficients manually, we can use the projection matrix from the class notes: $\hat{\beta} = (X^T X)^{-1} X^T Y$, which gives us the same values as $lm()$.

```
x.mat <- matrix(c(rep(1,nrow(snow)), snow[,2]), ncol = 2, nrow = 6)
y.mat <- as.matrix(snow[,3])

beta.hat <- solve(t(x.mat) %*% x.mat)%*%t(x.mat)%*%y.mat
beta.hat
```

```
##             [,1]
## [1,] -105.11466
## [2,]   34.25793
```

Question 2

$$
\begin{aligned}
Var(\hat{\alpha}) &= Var\left(\frac{\sum_{i=1}^{n}(x_i-\bar{x})y_i}{\sum_{i=1}^{n}(x_i-\bar{x})^2}\right) \\
&= Var\left(\frac{\sum_{i=1}^{n}(x_i-\bar{x})y_i}{S_{xx}}\right) \\
&= S_{xx}^{-2}Var\Big(\sum_{i=1}^{n}(x_i-\bar{x})y_i\Big) \\
&= S_{xx}^{-2}Var\Big(\sum_{i=1}^{n}(x_i-\bar{x})y_i)\Big) \\
&= S_{xx}^{-2}\sum_{i=1}^{n}(x_i-\bar{x})^2Var(y_i) \\
&= S_{xx}^{-2}(S_{xx})Var(y_i) \\
&= \frac{\sigma^2}{S_{xx}}
\end{aligned}
$$

$$
\begin{aligned}
Var(\hat{\mu}) &= Var(\bar{Y}-\hat{\alpha}\bar{X}) \\
&= Var(\bar{Y}) + \bar{X}^2Var(\hat{\alpha}) - 2\bar{x}Cov(\bar{Y},\hat{\mu})
\end{aligned}
$$

Now, we know from previous courses that $Var(\bar{Y}) = \sigma^2/n$ and we solved for $Var(\hat{\alpha})$ above, so we just need $Cov(\bar{Y},\hat{\mu})$.

$$
\begin{aligned}
Cov(\bar{Y},\hat{\mu}) &= Cov\Big(\frac{1}{n}\sum_{i=1}^{n}y_i, S_{xx}^{-1}\sum_{i=1}^{n}(x_i-\bar{x})y_i\Big) \\
&= \Big(\frac{1}{n}\Big)S_{xx}^{-1}Cov\Big(\sum_{i=1}^{n}y_i, \sum_{i=1}^{n}(x_i-\bar{x})y_i\Big) \\
&= \Big(\frac{1}{n}\Big)S_{xx}^{-1}\Big(\sum_{i=1}^{n}(x_i-\bar{x})\Big)Cov\Big(\sum_{i=1}^{n}y_i, \sum_{i=1}^{n}y_i\Big) \\
&= \Big(\frac{1}{n}\Big)S_{xx}^{-1}\Big(0\Big)Cov\Big(\sum_{i=1}^{n}y_i, \sum_{i=1}^{n}y_i\Big) \\
&= 0
\end{aligned}
$$

Therefore, we have $Var(\hat{\mu}) = Var(\bar{Y}) + \bar{X}^2Var(\hat{\alpha}) = \frac{\sigma^2}{n} + \bar{X}^2\Big(\frac{\sigma^2}{S_{xx}}\Big)$

Question 3

We compute $\hat{\sigma^2} = \frac{\sum_{i=1}^{N}(Y_i-\hat{Y}_i)^2}{n-2}$ in R using the fitted values from the above model as $\hat{Y}_i$ for $i \in \{1,\ldots,N\}$.

```
sigma.hat.sq <- sum((snow$avg_snow - m$fitted.values)^2)/(nrow(snow) - 2)
sigma.hat.sq
```

```
## [1] 1933.905
```

Question 4

We can use the $confint()$ function in R, giving it our model object.

```
confint(m, level = 0.95)
```

```
##                    2.5 %    97.5 %
## (Intercept) -346.5442215 136.31491
## elevation      0.4135908  68.10227
```

We confirm that these are the same values from the manual computation using a t-statistic. We implement the following formula for the confidence interval: $\hat{\beta} \pm t_{\alpha/2;n-p}\sqrt{\hat{var}(\beta)}$, where $\hat{var}(\beta) = \hat{\sigma^2}(X^T X)^{-1}$

```
n <- nrow(snow)
p <- ncol(x.mat)
alpha <- 0.05

t.stat <- qt(1-alpha/2, n - p)

covar.beta.hat <- sigma.hat.sq*solve(t(x.mat)%*%x.mat)

mu.upp <- beta.hat[1,1] + t.stat*sqrt(covar.beta.hat[1,1])
mu.low <- beta.hat[1,1] - t.stat*sqrt(covar.beta.hat[1,1])

alpha.upp <- beta.hat[2,1] + t.stat*sqrt(covar.beta.hat[2,2])
alpha.low <- beta.hat[2,1] - t.stat*sqrt(covar.beta.hat[2,2])

parameters <- c("Intercept", "Elevation")
lower <- c(round(mu.low, 3), round(alpha.low, 3))
upper <- c(round(mu.upp, 3), round(alpha.upp, 3))
cbind(parameters, lower, upper)
```

```
##      parameters  lower      upper
## [1,] "Intercept" "-346.544" "136.315"
## [2,] "Elevation" "0.414"    "68.102"
```

Clearly, we get the same confidence intervals through these two methods. We note that the interval for $/mu$ contains negative numbers. In this model μ is our intercept term, and therefore represents the fitted average annual snowfall for a town with elevatoin = 0 feet. Clearly, we cannot have negative snowfall, but this model is trained on data from Colorado, which doesn't have cities below 4K feet in elevation. Accordingly, our model is not appropraite for inference on cities at 0 feet in elevation, so the negative values for μ is not super problematic.

Question 5

We first write the covariance matrix $\sum$ as a matrix in R and then decompose it into its eigenvectors using the handy base R function $eigen()$.

```
sigma <- matrix(c(1, .5, .25, .5, 1, .5, .25, .5, 1),
                ncol = 3, byrow = TRUE)
sigma
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.5 0.25
## [2,] 0.50  1.0 0.50
## [3,] 0.25  0.5 1.00
```

```
eigen(sigma)
```

```
## eigen() decomposition
## $values
## [1] 1.8430703 0.7500000 0.4069297
##
## $vectors
##            [,1]          [,2]       [,3]
## [1,] -0.5417743 -7.071068e-01  0.4544013
## [2,] -0.6426206 -1.110223e-16 -0.7661846
## [3,] -0.5417743  7.071068e-01  0.4544013
```

Now we confirm that the P matrix is sufficiently close to the identity matrix.

```
p <- as.matrix(eigen(sigma)$vectors)
p%*%t(p) # this is close enough to the identity, so this is the correct matrix
```

```
##              [,1]          [,2]          [,3]
## [1,] 1.000000e+00  1.665335e-16  1.387779e-16
## [2,] 1.665335e-16  1.000000e+00 -5.551115e-17
## [3,] 1.387779e-16 -5.551115e-17  1.000000e+00
```

Now, we compute the D matrix accordingly and confirm that it is a nice diagonal matrix.

```
d <- p %*% sigma %*% t(p)
d # this is the diagonal matrix we want
```

```
##            [,1]       [,2]       [,3]
## [1,]  0.9386905  0.5288607 -0.1230915
## [2,]  0.5288607  1.2461830 -0.4673150
## [3,] -0.1230915 -0.4673150  0.8151265
```

To do this by hand, we set

$$\begin{aligned} 0 &= det(\sum -\lambda I_3) \\ &= (1-\lambda)((1-\lambda^2) - .25) - .5(.5(1-\lambda) - .125) + .25(.25 - .25(1-\lambda)) \\ &= (1-\lambda)^3 - .5625(1-\lambda) + .125 \end{aligned}$$

Since this is cubic with respect to λ, we know there can be 3 solutions to the equation, and there are many methods to find that these solutions are $\lambda_1 = 1.8430703, \lambda_2 = 0.75, \lambda_3 = 0.4069297$. R gives us a nice method to do this.

```
library(rootSolve)
jj <- function(x){(1-x)^3 - 0.5625*(1-x) + 0.125}
eigenvalues <- uniroot.all(jj, lower = 0, upper = 3)
eigenvalues
```

```
## [1] 0.7500000 0.4069255 1.8430856
```

We note that these are the same as the eigenvalues returned above by the *eigen*() function.

Now that we have our eigenvalues, we know our eigenvectors satisfy $(\Sigma - \lambda_i I_3)v_i = \vec{0_3}$, where v_i is a 1 x 3 vector for $i \in \{1, 2, 3\}$ and $\vec{0_3}$ is a 1 x 3 vecor with 0 as each entry. For each i, we solve the system of equations to get the 3 values for v_i. With access to a computer, we can alternatively solve $v_i = (\Sigma - \lambda_i I_3)^{-1}\mathbf{0}_3$. We list the eigenvectors in a matrix, with each column as an eigenvector.

```
eigen(sigma)$vectors
```

```
##            [,1]          [,2]       [,3]
## [1,] -0.5417743 -7.071068e-01  0.4544013
## [2,] -0.6426206 -1.110223e-16 -0.7661846
## [3,] -0.5417743  7.071068e-01  0.4544013
```

Question 6

We know that if a random variable is a linear transformation of another random variable, then their covariance matrix will have an eignevalue = 0. We generate a random vector x and a linear transformation y of that vector to demonstrate that their covariance matrix has an eignevalue of zero.

```
set.seed(112358)
x <- rnorm(100)
y <- 2*x + 2
df <- data.frame(x, y)
cov(df)
```

```
##          x        y
## x 0.9562805 1.912561
## y 1.9125611 3.825122
```

```
eigen(cov(df))
```

```
## eigen() decomposition
## $values
## [1] 4.781403 0.000000
##
## $vectors
##           [,1]       [,2]
## [1,] 0.4472136 -0.8944272
## [2,] 0.8944272  0.4472136
```

Question 7

In class, we reviewed the multivariate normal distribution and we know that $-Q/2 = (Y - \mu)^T V^{-1}(Y - \mu)$, where V^{-1} is a valid covariance matrix (symmetric and non-negative). We define

$$V^{-1} = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

Since $(Y-\mu)^T$ is a 1x3 vector, we know this computation $(Y-\mu)^T V^{-1}(Y-\mu)$ results in a 1x1 scalar. Now, it is easy enough to compute that

$$\begin{aligned}
Q &= 2y_1^2 + y_2^2 2 + y_3^2 + 2y_1y_2 - 8y_1 - 4y_2 + 8 \\
&= (Y-\mu)^T V^{-1}(Y-\mu) \\
&= (y_1-\mu_1)\Big(a(y_1-\mu_1) + b(y_2-\mu_2) + c(y_3-\mu_3)\Big) \\
&+ (y_2-\mu_2)\Big(b(y_1-\mu_1) + d(y_2-\mu_2) + e(y_3-\mu_3)\Big) \\
&+ (y_3-\mu_3)\Big(c(y_1-\mu_1) + e(y_2-\mu_2) + f(y_3-\mu_3)\Big)
\end{aligned}$$

Now we want to match coefficients, starting with the quadratic terms, it is easy enough to see that $a = 2$, $d = 1$ and $f = 1$ since these are the coefficients assigned to the y_1^2, y_2^2, and y_3^2 terms, respectively.

Now, we can note that our expression of Q does not include any y_1y_3 or y_2y_3 cross terms. This tells us that $c = e = 0$. We do have one corss term that we can match - we see Q includes a $2y_1y_2$ term, which must come from $2b(y_1-\mu_1)(y_2-\mu_2)$, and therefore $b = 1$

The next step is to plug in these values and expand the multiplication and match the non-quadratic, non-cross term coefficients. Skipping some algebra, we get the system of equations: $8y_i = -4y_i\mu_1$, $-4y_2 = -2y_2\mu_2 - 2\mu_1y_2$ and $8 = 2\mu_1^2 + 2\mu_1\mu_2 + \mu_2^2 + \mu_3^2$.

We can solve this system and get see that $\mu_1 = -2$, $\mu_2 = 4$ and $\mu_3 = 0$. This defines the vector $\vec{\mu}$ that we wanted to find. We also have solved for the covariance matrix:

$$V^{-1} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Question 8

Let matrices A and B be as defined in the problem and let them have entries $a_{i,j}$ and $b_{i,j}$, respectively. We know that AB is a m x m matrix and BA is a n x n matrix. Now, we note that there is no need to compute the full matrix multiplications, since we are intereted in the trace, we only need to concern ourselves with the diagonal entries.

Using the definition of matrix multiplication, we know that $AB_{k,k} = \sum_{i=1}^{n} a_{k,i} * b_{i,k}$ for $k \in \{1, \ldots, m\}$. Since the trace of a matrix is the sum of its diagonal entries, we know that $tr(AB) = \sum_{i=1}^{n} a_{1,i} * b_{i,1} + \sum_{i=1}^{n} a_{2,i} * b_{i,2} + \cdots + \sum_{i=1}^{n} a_{m,i} * b_{i,m} = \sum_{j=1}^{m}\sum_{i=1}^{n} a_{j,i} * b_{i,j}$.

By the same logic, we know that $BA_{k,k} = \sum_{i=1}^{m} b_{k,i} * a_{i,k}$ for $k \in \{1, \ldots, n\}$. It follows that

$$\begin{aligned}
tr(BA) &= \sum_{j=1}^{n}\sum_{i=1}^{m} b_{j,i} * a_{i,j} \\
&= \sum_{j=1}^{m}\sum_{i=1}^{n} a_{j,i} * b_{i,j} \\
&= tr(AB)
\end{aligned}$$

Question 9

We begin with the case $\vec{Y} \sim N_n(\vec{0}, \Sigma)$. First we note that $Y^T AY$ is a scalar, since $(1xn)x(nxn)x(nx1) = (1x1)$. Now,

$$\begin{aligned} E(tr(Y^T AY)) &= E(tr(AYY^T)) \\ &= tr(E(AYY^T)) \\ &= tr(AE(YY^T)) \\ &= tr(A\Sigma) \end{aligned}$$

Now, we want to generalize this for the case where $\vec{Y} \sim N_n(\vec{\mu}, \Sigma)$. To do this, we follow the same process as before, but now we know that $\Sigma = E[(\vec{y} - \vec{\mu})^T(\vec{y} - \vec{\mu})]$. So, we have:

$$\begin{aligned} E(tr((\vec{y} - \vec{\mu})^T A(\vec{y} - \vec{\mu}))) &= E(tr(A(\vec{y} - \vec{\mu})(\vec{y} - \vec{\mu})^T)) \\ &= tr(E(A(\vec{y} - \vec{\mu})(\vec{y} - \vec{\mu})^T)) \\ &= tr(AE((\vec{y} - \vec{\mu})(\vec{y} - \vec{\mu})^T)) \\ &= tr(A\Sigma) \end{aligned}$$

Question 10

To show that $M_2M_1 = M_1$, we need to show that $M_2M_1\vec{v} = M_1\vec{v} \forall \vec{v} \in \mathbf{V}$. Now, we know that $M_1\vec{v}$ is a vector in the column space of M_1, which means that $M_1\vec{v}$ is also in the column space of M_2 since M_2 contains all the elements of M_1. Since $M_1\vec{v}$ is in the column space of M_2, we know that transforming / multiplying $M_2M_1\vec{v} = M_1\vec{v}$.