DATA 609: Homework 5

Discrete Probabilistic Modeling

Aaron Grzasko

September 17, 2017

```r
library(dplyr)
library(tidyr)
library(knitr)
library(ggplot2)
library(ggthemes)
library(latex2exp)
library(igraph)
```

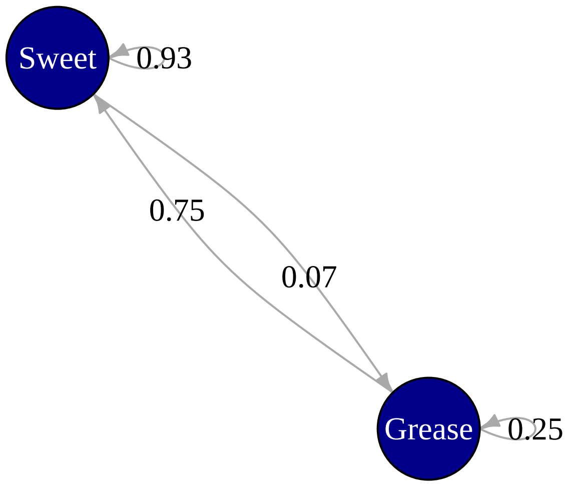
Page 228, Question 1

Consider a model for the long-term dining behavior of the students at College USA. It is found that 25% of the students who eat at the college's Grease Dining Hall return to eat there again, whereas those who eat at Sweet Dining Hall have a 93% return rate. These are the only two dining halls available on campus, and assume that all students eat at one of these halls. Formulate a model to solve for the long-term percentage of students eating at the hall

Model Formulation

Here is a network graph of the transition probabilities:

```r
# create network graph
mygraph <- graph(c("Sweet","Sweet","Sweet","Grease","Grease","Sweet",
                   "Grease","Grease"))
E(mygraph)$label <- c(0.93,0.07,0.75,0.25)

plot(mygraph, edge.arrow.size=.5, vertex.color="darkblue", vertex.size=55,
     vertex.frame.color="black", vertex.label.color="white",
     vertex.label.cex=1.0, vertex.label.dist=0.0, edge.curved=0.2,
     edge.label.cex=1.0, edge.color = 'darkgray', edge.label.color = "black")
```



Below is the same information presented in table form:

```
# create table
mydf <- data.frame(Present_State = c("Grease","Sweet"),Grease_Next=c(0.25,0.07),
                   Sweet_Next=c(0.75,0.93))
kable(mydf)
```

Present_State	Grease_Next	Sweet_Next
Grease	0.25	0.75
Sweet	0.07	0.93

Now let's state the problem as a system of linear equations:

G_n =the percentage of students eating at Greasy Dining Hall in period n

S_n =the percentage of students eating at Sweet Dining Hall in period n

$$G_{n+1} = 0.25G_n + 0.07S_n$$
$$S_{n+1} = 0.75G_N + 0.93S_n$$

We can also represent this system using a transition matrix:

$$A = \begin{bmatrix} 0.25 & 0.07 \\ 0.75 & 0.93 \end{bmatrix}$$

Let

$$\vec{x_n} = \begin{bmatrix} G_n \\ S_n \end{bmatrix}$$

We can solve for the percentage of diners at time n for each location,$\vec{x_n}$, using the following formula:

$$\vec{x_n} = A\ \vec{x_{n-1}} = A^n\ \vec{x_0}$$

Long Term Behavior

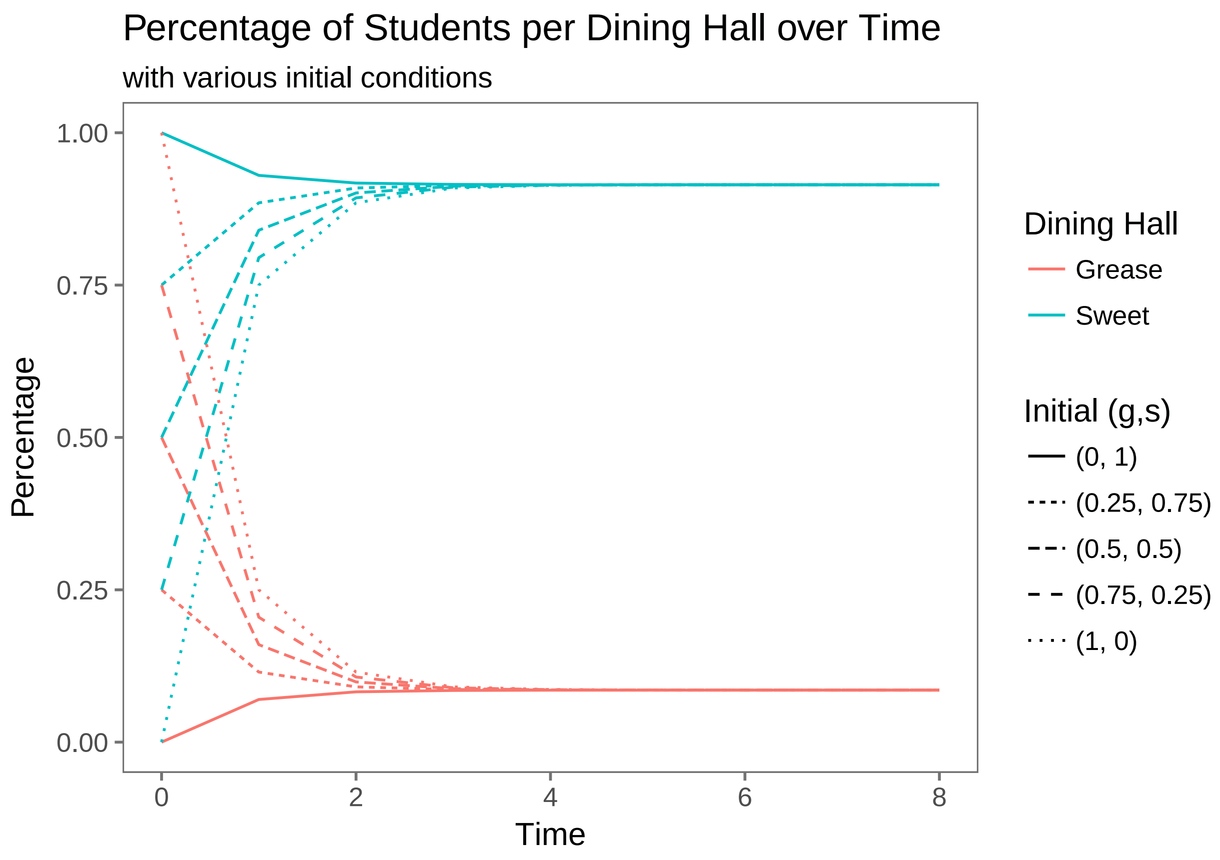
Using the equation above, we can model the long-term behavior of students' dining hall preferences.

In the following scripts, we model the percentage of students at each dining common for times 0 through 8. We also model five different initial assumptions for x_0.

```r
# number of periods to run
n <- 8

# transition matrix
A <- matrix(c(0.25,0.75,0.07,0.93),2,2)

# cycle through a variety of initial Grease values
g0 <- c(0,0.25, 0.5, 0.75, 1.0)

# master data frame
mydf <- data.frame(Time=integer(), Grease=numeric(), Sweet=numeric(), Initial=character())

# loop through g0 starting values
for (i in g0){

    # Sweet percentage must be 1 - Grease percentage
    s0 <- 1 - i

    # assign initial vector x
    x <- c(i,s0)
    initial <- paste0("(",i,", ",s0,")")

    # grease = x1; sweet = x2;start at time 0
    g <- x[1]
    s <- x[2]
    t <- 0

    # loop through n periods
    for (j in 1:n) {
        x <- A %*% x   # calculate percentages in each diner at time n
        g <- append(g,x[1])
        s <- append(s,x[2])
        t <- append(t,j)
        initial <- append(initial, paste0("(",i,", ",s0,")"))
    }
    # temporary data frame
    tempdf <- data.frame(Time=t,Grease=g,Sweet=s, Initial=initial)

    # append to master data frame
    mydf <- rbind(mydf,tempdf)
}
```

Here is a plot of our model:

```r
# transform data frame for easy plotting
mydf2 <- mydf %>%
    gather("Dining_Hall","Percentage",c("Grease","Sweet"))

# plot
plt <- ggplot(data=mydf2,aes(x=Time,y=Percentage,col=Dining_Hall)) +
  geom_line(aes(linetype = Initial)) +
  labs(title = "Percentage of Students per Dining Hall over Time",
       subtitle="with various initial conditions", col = "Dining Hall",
       linetype = "Initial (g,s)") + theme_few()

plt
```



From this visualization, we see that the percentage of eaters at each dining hall converges to a stable solution after approximately four periods.

Regardless of the initial conditions, the percentage of students eating at Greasy and Sweet diners after 8 periods are roughly 8.5% and 91.5%, respectively–see below:

```r
mydf <- mydf %>%
    filter(Time==8) %>%
    select(Initial, Grease, Sweet)
```

```r
kable(mydf)
```

Initial	Grease	Sweet
(0, 1)	0.0853658	0.9146342
(0.25, 0.75)	0.0853660	0.9146340
(0.5, 0.5)	0.0853663	0.9146337
(0.75, 0.25)	0.0853666	0.9146334
(1, 0)	0.0853669	0.9146331

Check Work

Let P be a 2 x 2 matrix representing the eigenvectors of matrix A:

```r
# calculate eigenvectors
P <- eigen(A)$vectors
P
```

```
            [,1]       [,2]
[1,] -0.09292945 -0.7071068
[2,] -0.99567270  0.7071068
```

Now, calculate diagonal matrix D as follows:

$$D = P^{-1}AP$$

```r
# calculate diagonal matrix, D
D <- solve(P) %*% A %*% P
D
```

```
              [,1] [,2]
[1,]  1.000000e+00 0.00
[2,] -2.428613e-17 0.18
```

With additional manipulation, we have

$$A^n = PD^nP^{-1}$$

For large n:

$$\lim_{n\to\infty} D^n = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

and A^n is approximately:

```r
# Calculated A^n as n approaches infinity

D_mod <- matrix(c(1,0,0,0),2,2)

A_to_n <- P %*% D_mod %*% solve(P)
A_to_n
```

```
           [,1]       [,2]
[1,] 0.08536585 0.08536585
[2,] 0.91463415 0.91463415
```

Now, for an arbitrary 2 x 1 vector $\vec{x}$ with components χ_0, and $\chi_1 = 1 - \chi_0$:

$$\lim_{n\to\infty} A^n \vec{x} = \lim_{n\to\infty} A^n \begin{bmatrix} \chi_0 \\ 1-\chi_0 \end{bmatrix} = \chi_0 \begin{bmatrix} 0.0854 \\ 0.9146 \end{bmatrix} + (1-\chi_0) \begin{bmatrix} 0.0854 \\ 0.9146 \end{bmatrix} = \begin{bmatrix} 0.0854 \\ 0.9146 \end{bmatrix}$$

This solution is consistent with our previously modeled result.

Page 232: Question 1

Consider a stereo with CD player, FM-AM radio tuner, speakers (dual), and power amplifier (PA) components, as displayed with the reliabilities shown below. Determine the system's reliability. What assumptions are required in your model

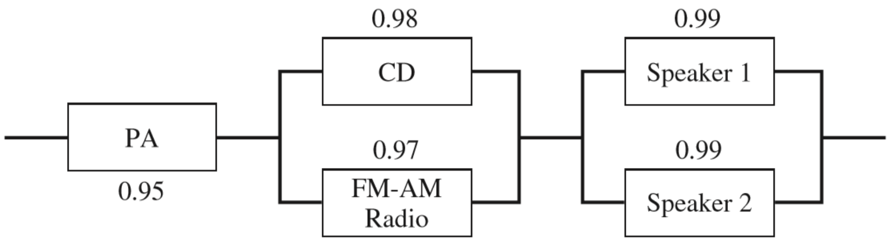


Figure 1:

Define $F(t)$ as the cumulative distribution function of the failure rate, $f(t)$, of a system or component, where t refers to time. The reliability, $R(t)$, of a system of component is the complement of $F(t)$:

$$R(t) = 1 - F(t)$$

```r
# data
R_i <- c(0.95, 0.98, 0.97, 0.99, 0.99)
names(R_i) <- c("pa","cd","radio","speaker1","speaker2")

kable(R_i)
```

pa	0.95
cd	0.98
radio	0.97
speaker1	0.99
speaker2	0.99

Assumptions

The diagram above assumes the following subsystems perform in parallel (i.e. all components in the subsystem must fail before system failure):

- Subsystem 1: CD and FM/AM Radio
- Subsystem 2: Speaker 1 and Speaker 2

These assumptions seem somewhat reasonable, albeit very simplistic:

- Subsystem 1 refers to devices that read, interpret, and play back music and other audio data. Given that a listener has no personal preference of listening to music on CD vs. radio music, then this assumption is reasonable.

- Subsystem 2 refers to speakers, which are hardware devices that generates sound. If a listener is not concerned with audio quality (e.g. the ability to hear music in stereo), then this assumption might be reasonable.

The diagram also indicates that the parallel subsystems run in serial fashion with each other and the power amplifier.

In other words, we assume the system as a whole fails when any of the two subsystems or the PA component fails.

These assumptions are reasonable:

- A stereo needs a power amplifier to strengthen the low-power electronic signal from the radio or CD player and drive the speakers. Without this component, no sounds are generated.
- A proper music device (radio or CD player) is necessary for interpreting audio signal input and sending audio output.
- A stereo cannot function without at least one speaker to produce sound.

Without additional guidance, we assume that the individual components within each parallel subsystem are independent of each other. Finally, we assume that all components and subsystems running in serial fashion have independent reliabilities.

Reliability Calculation

Reliability for a parallel system, $R_s(t)$ is calculated as the union of the reliability for the individual components:

$$R_s(t) = \bigcup_{i=1}^{n} R_i(t)$$

Let's calculate the reliabilities for each of the three parallel subsystems:

```r
# Reliability of || subsystem 1
R_ps1 <- R_i["cd"] + R_i["radio"] -  R_i["cd"] * R_i["radio"]
names(R_ps1) <- NULL
R_ps1
```

```
[1] 0.9994
```

```r
# Reliability of || subsystem 2
R_ps2 <- R_i["speaker1"] + R_i["speaker2"] -  R_i["speaker1"] * R_i["speaker2"]
names(R_ps2) <- NULL
R_ps2
```

```
[1] 0.9999
```

For serial systems with independent components, the reliability, $R_s(t)$, is calculated as follows:

$$R_s(t) = \prod_{i=1}^{n} R_i(t)$$

We can now solve for the reliability of the total system, assuming the PA runs in serial with the two subsystems:

```r
# calculate reliability of total system
R_s <- R_i["pa"] * R_ps1 * R_ps2
names(R_s) <- NULL
R_s
```

```
[1] 0.9493351
```

Based on our model, the stereo system has a 94.9% probability of functioning properly.

Page 240: Question 1

Use the basic linear model $y = b_1 x + b_0$ to fit the following data sets. Provide the model; provide the values of SSE, SSR, SST, and R^2; and provide a residual plot.

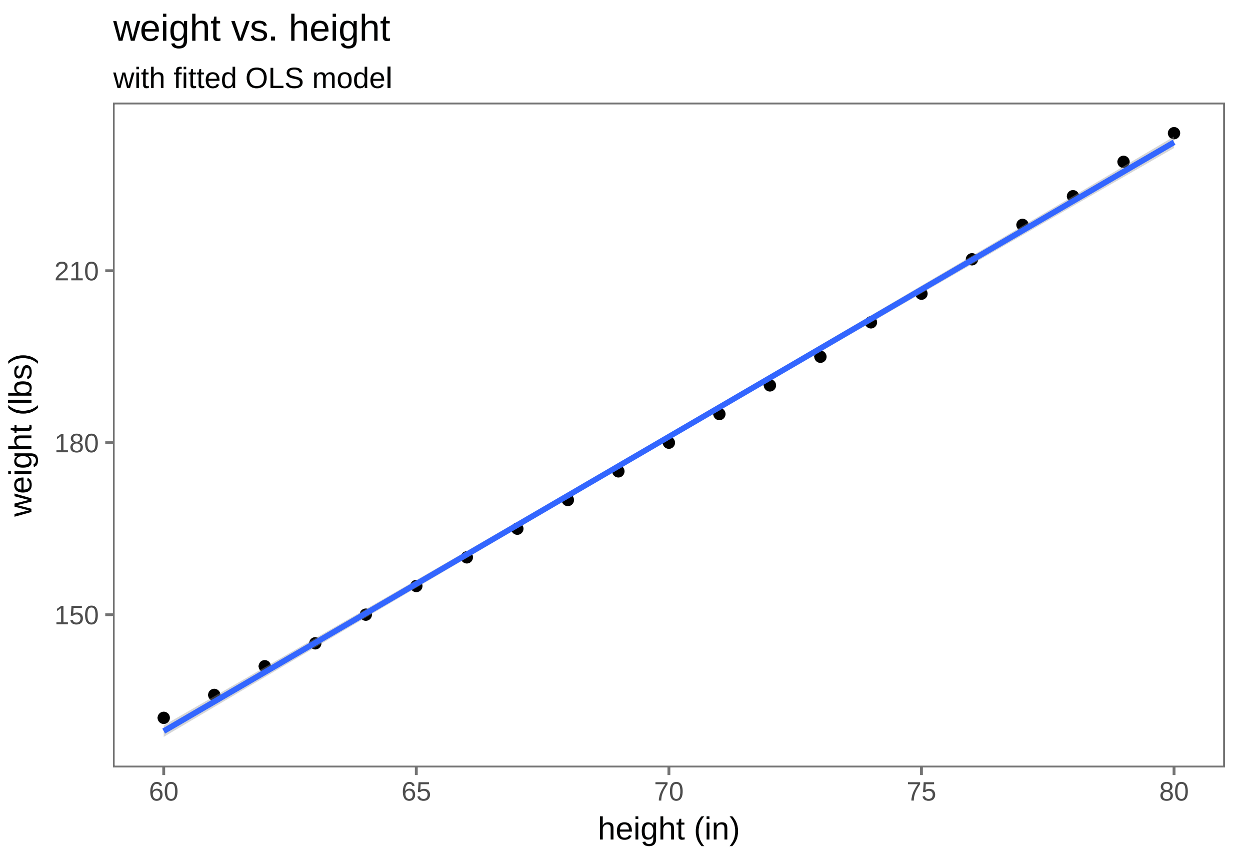
For Table 2.7, predict the weight as a function of height.

```
# data
ht <- 60:80
wt <- c(132,136,141,145,150,155,160,165,170,175,180,185,190,195,201,206,212,218,223,229,234)

mydf <- data.frame(height=ht, weight=wt)
kable(mydf)
```

height	weight
60	132
61	136
62	141
63	145
64	150
65	155
66	160
67	165
68	170
69	175
70	180
71	185
72	190
73	195
74	201
75	206
76	212
77	218
78	223
79	229
80	234

Let's produce a scatterplot of the data:

```
# plot
g <- ggplot(mydf, aes(x=height,y=weight)) + geom_point() + theme_few() +
  labs(title="weight vs. height", x="height (in)",y="weight (lbs)",
           subtitle="with fitted OLS model") +
   geom_smooth(method="lm")
g
```



Based on the scatterplot output, we suspect that linear relationship between height and weight is very strong.

Find OLS coefficients

Let's solve for the regression coefficients using derived equations:

$$b_1 = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

```r
# find slope coefficient, b1
b1 <- (nrow(mydf) * sum(mydf$height * mydf$weight) - sum(mydf$height) * sum(mydf$weight)) /
    (nrow(mydf) * sum(mydf$height^2) - sum(mydf$height)^2)
b1
```

```
[1] 5.136364
```

$$b_0 = \bar{y} - b_1\bar{x}$$

```r
# find intercept, b0
b0 <- mean(mydf$weight) - b1 * mean(mydf$height)
b0
```

```
[1] -178.4978
```

Now, let's solve for the coefficients using linear algebra:

$$\vec{b} = (X^T X)^{-1} X^T \vec{y}$$

```r
# X Matrix
X <- cbind(rep(1,nrow(mydf)), mydf$height)

# y vector
y <- wt

# solve for coefficient vector b
b <- solve(t(X) %*% X) %*% t(X) %*% y
b
```

```
             [,1]
[1,] -178.497835
[2,]    5.136364
```

Finally, let's check our work using the `lm()` function in the stats package:

```r
mylm <- lm(weight~height, data=mydf)
mylm$coefficients
```

```
(Intercept)      height 
-178.497835    5.136364 
```

SSE, SST, SST, and R-Squared

Find SSE:

$$SSE = \sum_{i=1}^{m} [y_i - (b_1 x_i + b_0)]^2$$

```r
sse <- sum((mydf$weight -  (b1*mydf$height + b0))^2)
sse
```

```
[1] 24.6342
```

Find SST:

$$SST = \sum_{i=1}^{m} (y_i - \bar{y})^2$$

```r
sst <- sum((mydf$weight - mean(mydf$weight))^2)
sst
```

```
[1] 20338.95
```

Find SSR:

$$SSR = SST - SSE$$

```r
ssr <- sst - sse
ssr
```

```
[1] 20314.32
```

Find R^2:

$$R^2 = 1 - \frac{SSE}{SST}$$

```r
r2 <- 1 - sse/sst
r2
```

```
[1] 0.9987888
```

Let's check our R^2 calculation using the `lm()` function:

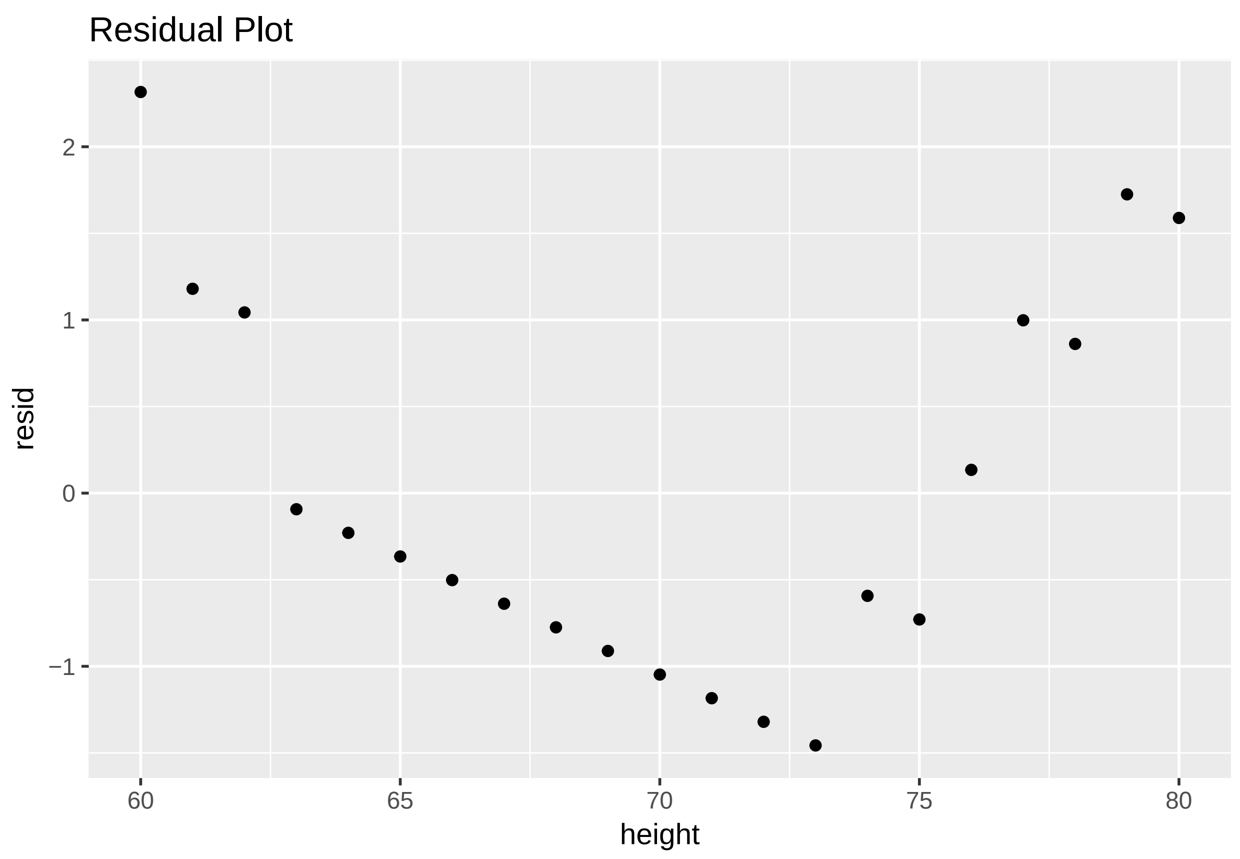
```r
summary(mylm)$r.squared
```

```
[1] 0.9987888
```

The high R^2 value, as seen here, is generally associated with a strong model fit.

Residual Plot

```r
# add residuals to dataframe
mydf$resid <- mylm$residuals

# plot
ggplot(mydf, aes(x=height,y=resid)) + geom_point() +
    labs(title = "Residual Plot")
```



The residuals do not appear to be distributed in a random fashion–one of the key assumptions of simple OLS regression. Therefore, we must be cautious in interpreting the regression output.

Page 240: Question 2

Use the basic linear model $y = b_1x + b_0$ to fit the following data sets. Provide the model; provide the values of SSE, SSR, SST, and R^2; and provide a residual plot.

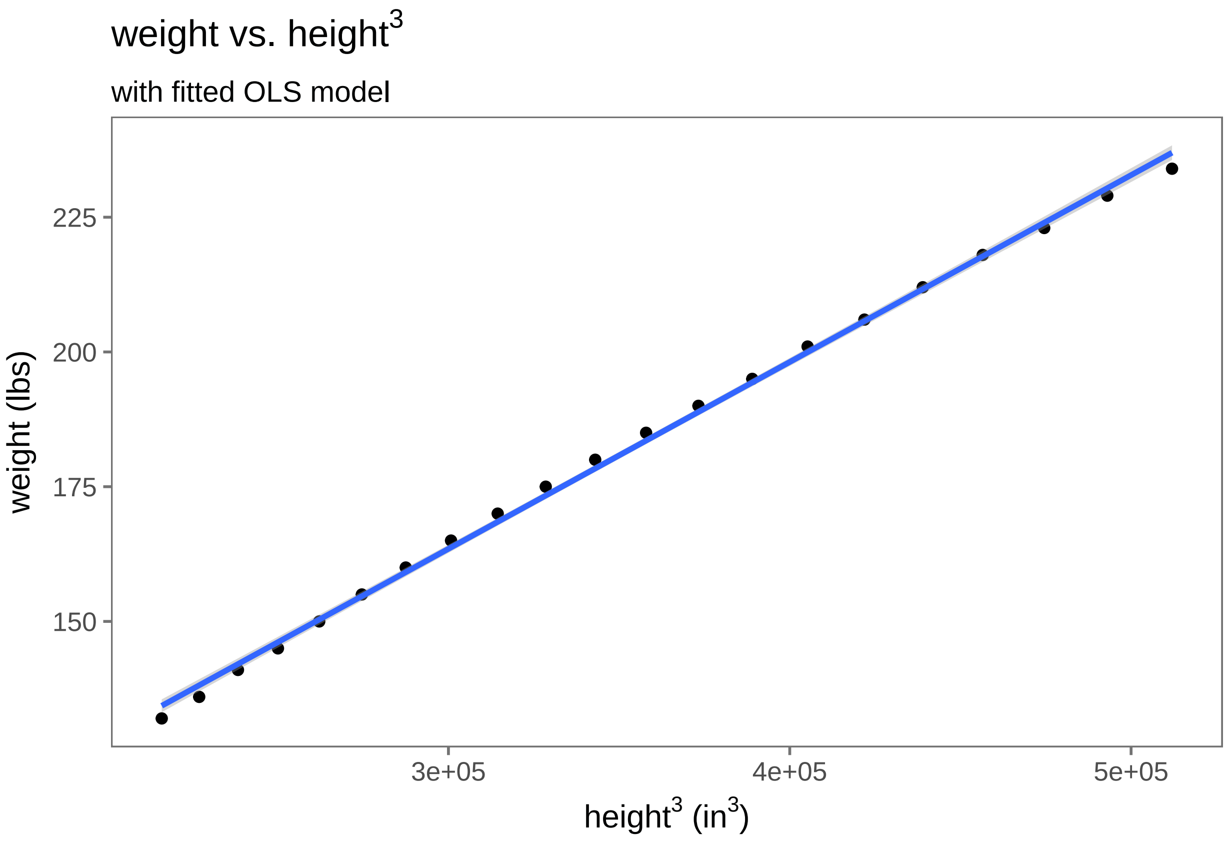
For Table 2.7, predict the weight as a function of the cube of the height.

```
# data
ht <- 60:80
wt <- c(132,136,141,145,150,155,160,165,170,175,180,185,190,195,201,206,212,218,223,229,234)
ht_cubed <- ht^3
mydf <- data.frame(height_cubed=ht_cubed, weight=wt)
kable(mydf)
```

height_cubed	weight
216000	132
226981	136
238328	141
250047	145
262144	150
274625	155
287496	160
300763	165
314432	170
328509	175
343000	180
357911	185
373248	190
389017	195
405224	201
421875	206
438976	212
456533	218
474552	223
493039	229
512000	234

Let's produce a scatterplot of the data:

```
# plot
g <- ggplot(mydf, aes(x=height_cubed,y=weight)) + geom_point() + theme_few() +
  labs(title=TeX("weight vs. $height^3$"), x=TeX("height^3 (in^3)"),y="weight (lbs)",
           subtitle="with fitted OLS model") +
   geom_smooth(method="lm")
g
```



Based on the scatterplot output, we suspect that linear relationship between height cubed and weight is very strong.

Find OLS coefficients

Let's solve for the regression coefficients using derived equations:

$$b_1 = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

```
# find slope coefficient, b1
b1 <- (nrow(mydf) * sum(mydf$height_cubed * mydf$weight) - sum(mydf$height_cubed) * sum(mydf$weight)) /
    (nrow(mydf) * sum(mydf$height_cubed^2) - sum(mydf$height_cubed)^2)
b1
```

```
[1] 0.0003467044
```

$$b_0 = \bar{y} - b_1\bar{x}$$

```
# find intercept, b0
b0 <- mean(mydf$weight) - b1 * mean(mydf$height_cubed)
b0
```

```
[1] 59.4584
```

Now, let's solve for the coefficients using linear algebra:

$$\vec{b} = (X^T X)^{-1} X^T \vec{y}$$

```r
# X Matrix
X <- cbind(rep(1,nrow(mydf)), mydf$height_cubed)

# y vector
y <- wt

# solve for coefficient vector b
b <- solve(t(X) %*% X) %*% t(X) %*% y
b
```

```
             [,1]
[1,] 5.945840e+01
[2,] 3.467044e-04
```

Finally, let's check our work using the `lm()` function in the stats package:

```r
mylm <- lm(weight~height_cubed, data=mydf)
mylm$coefficients
```

```
 (Intercept) height_cubed
5.945840e+01 3.467044e-04
```

SSE, SST, SST, and R-Squared

Find SSE:

$$SSE = \sum_{i=1}^{m}[y_i - (b_1 x_i + b_0)]^2$$

```r
sse <- sum((mydf$weight -  (b1*mydf$height_cubed + b0))^2)
sse
```

```
[1] 39.86196
```

Find SST:

$$SST = \sum_{i=1}^{m}(y_i - \bar{y})^2$$

```r
sst <- sum((mydf$weight - mean(mydf$weight))^2)
sst
```

```
[1] 20338.95
```

Find SSR:

$$SSR = SST - SSE$$

```r
ssr <- sst - sse
ssr
```

```
[1] 20299.09
```

Find R^2:

$$R^2 = 1 - \frac{SSE}{SST}$$

```r
r2 <- 1 - sse/sst
r2
```

```
[1] 0.9980401
```

Let's check our R^2 calculation using the `lm()` function:

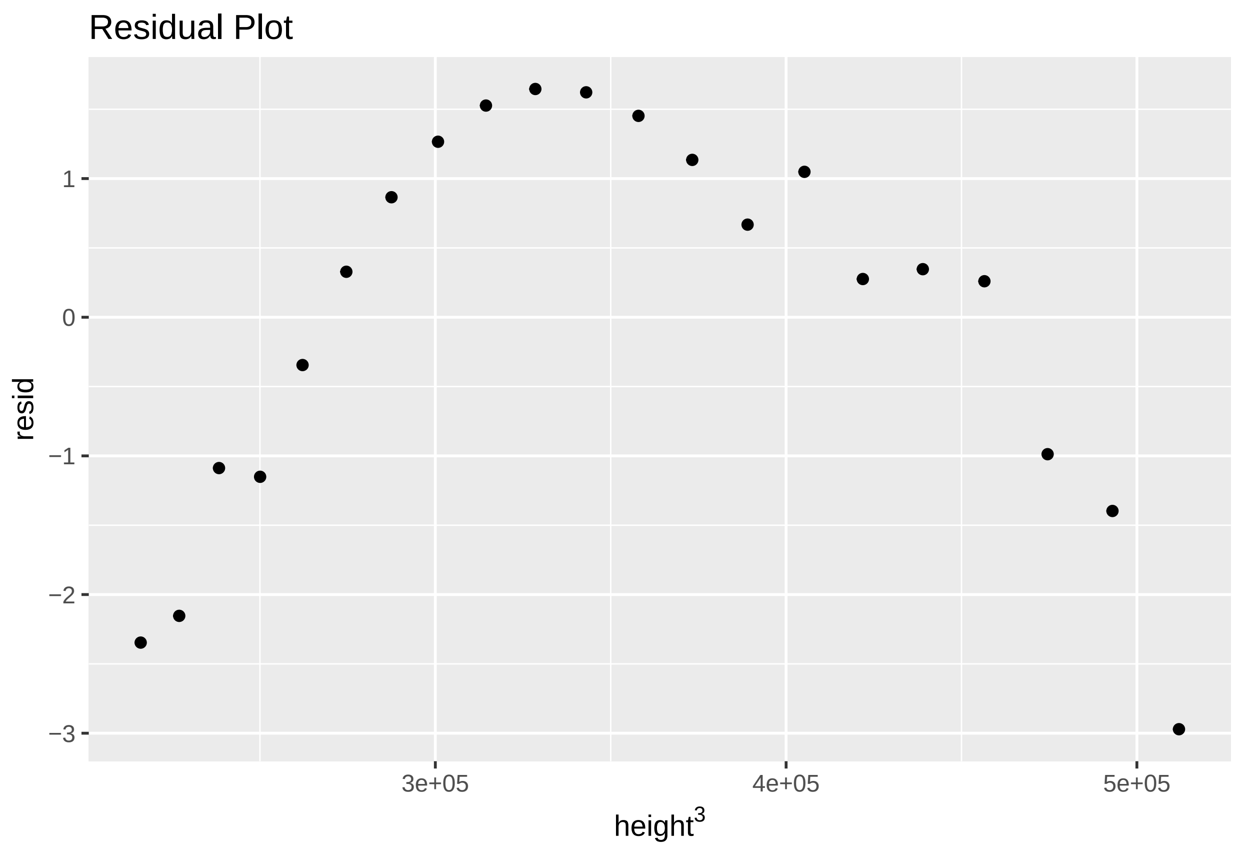
```r
summary(mylm)$r.squared
```

```
[1] 0.9980401
```

Once again, we see a high R^2 value, which is typically associated with a strong model fit.

Residual Plot

```r
# add residuals to dataframe
mydf$resid <- mylm$residuals

# plot
ggplot(mydf, aes(x=height_cubed,y=resid)) + geom_point() +
    labs(title = "Residual Plot", x=TeX("$height^3$"))
```



As in our previous model, the residuals do not appear to be distributed in a random fashion–one of the key assumptions of simple OLS regression. The pattern here is more distinct compared to our previous model. Given this pattern, we suspect that the previous, simpler model may be more appropriate.

References

-Long term behavior of Markov Chain models: http://www.sosmath.com/matrix/markov/markov.html