DATA 609: Final

Daina Bouquin, Christophe Hunt, Christina Taylor

May 15, 2017

Contents

1 Textbook Information

Giordano, F. R., Fox, W. P., & Horton, S. B. (2014). *A first course in mathematical modeling*. Australia: Brooks/Cole.

2 Textbook Part I:

2.1 The Problem

A farmer has 30 acres on which to grow tomatoes and corn. Each 100 bushels of tomatoes require 1000 gallons of water and 5 acres of land. Each 100 bushels of corn require 6000 gallons of water and 2.5 acres of land. Labor costs are $1 per bushel for both corn and tomatoes. The farm has available 30,000 gallons of water and $750 in capital. He knows he cannot sell more than 500 bushels of tomatoes and 475 bushels of corn. He estimates a profit of $2 on each bushel of tomatoes and $3 on each bushel of corn.

2.2 How many bushels should he raise to maximize profits?

Let t = bushels of tomatoes (in hundreds) c = bushels of corn (in hundreds) w = gallons of water (in thousands) p = profit (in dollars) a = land (in acres)

The farmer's problem can be modeled as Maximizing Profit: $200t + 300c$

Subject to the following constraints: Non-negativity: $t \geq 0, c \geq 0$ Water: $t + 6c \leq 30$ Land: $5t + 2.5c \leq 30$ Labor cost: $t + c \leq 7.5$ Demand: $t \leq 5, c \leq 4.75$

2.3 Apply linear programming solution

```
#create LP object with 2 decision variables
lprec = make.lp(0,2)
#set objective function and direction (maximize)
set.objfn(lprec,c(200,300))
lp.control(lprec,sense='max')
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] 1e+30
##
## $epsilon
##       epsb       epsd       epsel      epsint epsperturb   epspivot
```

```
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

```
#add constraints
add.constraint(lprec, c(1,6), "<=", 30)
add.constraint(lprec, c(5,2.5), "<=", 30)
add.constraint(lprec, c(1,1), "<=", 7.5)
#set boundaries
set.bounds(lprec, lower = c(0,0), upper = c(5,4.75))

#rename for printing
RowNames <- c("water", "land", "labor")
```

```
ColNames <- c("tomatoes","corn")
dimnames(lprec) <- list(RowNames, ColNames)
#verify the model
lprec
```

```
## Model name:
##            tomatoes      corn
## Maximize        200       300
## water             1         6  <=   30
## land              5       2.5  <=   30
## labor             1         1  <=  7.5
## Kind            Std       Std
## Type           Real      Real
## Upper             5      4.75
## Lower             0         0
```

```
#solve the model (0 means there is a feasible solution)
solve(lprec)
```

```
## [1] 0
```

```
#find extreme point
get.variables(lprec)
```

```
## [1] 3.0 4.5
```

```
#find maximize objective function value
get.objective(lprec)
```

```
## [1] 1950
```

We can see that if the farmer grows 300 bushels of tomatoes and 450 bushels of corn, he can maximize his profit at $1950.

2.4 Should he sign the contract?

Assume the farmer has the opportunity to sign a contract with grocery store to grow and deliver at least 300 bushels of tomatoes and 500 bushels of corn.

The new demand constraint is now $t \geq 3, c \geq 5$ Intuitively, we can easily see that this condition is incompatible with the constraint of labor costs. We can verify this using linear programming:

```
lprec = make.lp(0,2)
#set objective function and direction (maximize)
set.objfn(lprec,c(200,300))
lp.control(lprec,sense='max')
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
```

```
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] 1e+30
##
## $epsilon
##       epsb       epsd      epsel     epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
```

```
## [1] 0
##
## $verbose
## [1] "neutral"
```

```
#add constraints
add.constraint(lprec, c(1,6), "<=", 30)
add.constraint(lprec, c(5,2.5), "<=", 30)
add.constraint(lprec, c(1,1), "<=", 7.5)
#set boundaries
set.bounds(lprec, lower = c(3,5))

#solve the model
solve(lprec)
```

```
## [1] 2
```

As shown, there is no feasible solution. The farmer should not sign the contract, because realistically, he cannot deliver.

2.5 Should he obtain the additional water?

Assume the farmer can obtain an additional 10,000 gallons of water for $50.

The new water constraint is now $t + 6c \leq 40$ The farmer's profit is $200t + 300c - 50$

```
lprec = make.lp(0,2)
#set objective function and direction (maximize)
set.objfn(lprec,c(200,300))
lp.control(lprec,sense='max')
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] 1e+30
##
## $epsilon
##       epsb       epsd      epsel      epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12       1e-07      1e-05      2e-07
```

```
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

```r
#add constraints
add.constraint(lprec, c(1,6), "<=", 40)
add.constraint(lprec, c(5,2.5), "<=", 30)
add.constraint(lprec, c(1,1), "<=", 7.5)
#set boundaries
set.bounds(lprec, lower = c(0,0), upper = c(5,4.75))

RowNames <- c("water", "land", "labor")
ColNames <- c("tomatoes","corn")
dimnames(lprec) <- list(RowNames, ColNames)
```

```r
#solve the model
solve(lprec)
```

```
## [1] 0
```

```r
#find extreme point
get.variables(lprec)
```

```
## [1] 2.75 4.75
```

```r
#find maximize profit
get.objective(lprec) - 50
```

```
## [1] 1925
```

If the farmer grows 275 bushels of tomatoes and 475 bushels of corn, he can maximize his profit at $1975. However, given the $50 cost in acquiring extra water, his net profit actually went down to $1925. Therefore, he should not obtain additional water.

3 Textbook Part II:

Chapter 5.3, Project 1

3.1 The problem

Construct and perform a Monte Carlo simulation of blackjack.
Parameters:

- Play 12 games (simulations) where each game lasts two decks. When the two decks are out, the round is completed using two fresh decks (that is the last round of the game). Everything is then reset for the start of the next game.
- The dealer cannot see the players cards and vice versa.
- The player wins 3 dollars with a winning hand.
- The player loses 2 dollars with a losing hand.
- No money is exchanged if there is no winner.
  - There is no winner when neither has gone bust and they stand at the same amount.
  - If the dealer goes bust, the player automatically wins.
- The dealer strategy is to stand at 17 or above.
- The player strategy is open and can be set as desired.

3.2 Create a two deck game

The below functions create the game of blackjack from a dealt card, to a hand, to a round, and to a full two-deck game. Game results are stored as a vector: 1 = dealer win; 2 = player win; 0 = no winner. The resulting vector is used to calculate player winnings/losses.

3.2.1 Create a hand for the dealer.

```
deal_card <- function(d_deck,decks_2){
  flag_last_round <- 0
  if (length(d_deck) == 0) {
    flag_last_round <- 1
    d_deck <- decks_2
  }
  d_card <- sample(d_deck,1,FALSE)
  d_deck <- d_deck [!d_deck %in% d_card | duplicated(d_deck)]
  return(list(d_card,d_deck,flag_last_round))
}
dealer_hand <- function(dh_deck,decks_2) {
  dh_hand <- 0
  dh_last_round_flag <- 0
  dh_count <- 1
```

```
  while (sum(dh_hand) < 17) {
    dh_deal <- deal_card(dh_deck,decks_2)
    dh_hand[dh_count] <- dh_deal[[1]]
    dh_deck <- dh_deal[[2]]
    if (dh_deal[[3]] == 1) {dh_last_round_flag <- dh_deal[[3]]}
    dh_count <- dh_count+1 # ace = 1 if hand > 21
    if(sum(dh_hand) > 21) {
      if(11 %in% dh_hand) {
        dh_hand[match(11,dh_hand)] <- 1
      }
    }
  }
  return(list(dh_hand,dh_deck,dh_last_round_flag))
}
```

3.2.2 Create player hand

```
player_hand <- function(ph_deck,decks_2,threshold_p) {
  ph_hand <- 0
  ph_count <- 1
  ph_last_round_flag <- 0
  while (sum(ph_hand) < threshold_p) {
    ph_deal <- deal_card(ph_deck,decks_2)
    ph_hand[ph_count] <- ph_deal[[1]]
    ph_deck <- ph_deal[[2]]
    if (ph_deal[[3]] == 1) {ph_last_round_flag <- ph_deal[[3]]}
    ph_count <- ph_count+1 # ace = 1 if hand > 21
    if(sum(ph_hand) > 21) {
      if(11 %in% ph_hand) {
        ph_hand[match(11,ph_hand)] <- 1
      }
    }
  }
  return(list(ph_hand,ph_deck,ph_last_round_flag))
}
```

3.2.3 Create a game

```
game <- function(game_deck,decks_2,threshold_p) {
  #dealer hand
  d_result <- dealer_hand(game_deck,decks_2)
  d_hand <- d_result[[1]]
  game_deck <- d_result[[2]]
  last_round_flag_d <- d_result[[3]]
  #player hand
  p_result <- player_hand(game_deck,decks_2,threshold_p)
  p_hand <- p_result[[1]]
  game_deck <- p_result[[2]]
  last_round_flag_p <- p_result[[3]]
  if (last_round_flag_d == 1 || last_round_flag_p == 1) {
```

```
    game_last_round_flag <- 1
  } else {
    game_last_round_flag <- 0
  }
  return(list(d_hand,p_hand,game_deck,game_last_round_flag))
}

# Go through two decks- create dealer and player hand each round
play_decks_2 <- function(p2d_deck,decks_2,threshold_p) {
  round_counter <- 1
  p2d_last_round_flag <- 0
  result_list.names <- c("Dealer_Hand", "Player_Hand")
  result_list <- vector("list", length(result_list.names))
  names(result_list) <- result_list.names
  while (p2d_last_round_flag != 1) {
    round_result <- game(p2d_deck,decks_2,threshold_p)
    result_list$Dealer_Hand[round_counter] <- list(round_result[[1]])
    result_list$Player_Hand[round_counter] <- list(round_result[[2]])
    p2d_deck <- round_result[[3]]
    p2d_last_round_flag <- round_result[[4]]
    round_counter <- round_counter+1
  }
  return(result_list)
}

# Takes in list containing all hands after above running through two decks.
results_vec <- function(results_set) {
  winner <- numeric(length(results_set$Dealer_Hand))
  for (i in 1:length(results_set$Dealer_Hand)) {
    d_Hand <- sum(results_set$Dealer_Hand[[i]])
    p_Hand <- sum(results_set$Player_Hand[[i]])
    if(d_Hand < 22) {
      if(p_Hand < 22) {
        if(d_Hand > p_Hand) {
          winner[i] <- 1
        } else if (d_Hand < p_Hand) {
            winner[i] <- 2
        } else if (d_Hand == p_Hand) {
            winner[i] <- 0
        }
      } else if(p_Hand >= 22) {
          winner[i] <- 1
      }
    } else if(d_Hand >= 22){
          winner[i] <- 2 # if dealer goes bust player wins
      }
  }
  return(winner)
}
```

3.3 Calculate the player winnings with assumptions

```
# player bets $2/hand
# player wins = player receives $3
# player loss = player loses $2 bet
# tie = $0 exchanged
# dealer goes bust = win for player
calc_player_win <- function(results) {
  winnings <- numeric(length(results))
  for(i in 1:length(results)) {
    if (results[i] == 0) winnings[i] <- 0
    if (results[i] == 1) winnings[i] <- -2
    if (results[i] == 2) winnings[i] <- 3
  }
  return(winnings)
}
```

3.4 Play a full two deck game, calculate winnings/losses

```
play_deck2 <- function(pd2_deck,threshold_p){
  #create list of results per round after running two decks game
  one_set <- play_decks_2(pd2_deck,pd2_deck,threshold_p)
  #create vector for results of each hand
  outcome_vec <- results_vec(one_set)
  player_winnings <- 0
  player_winnings <- calc_player_win(outcome_vec)
  return(sum(player_winnings))
}

# Create function to run the simulation for two deck games
# Takes two decks and number of simulations as params
# Each run uses player strategies (stand at 16 to 20); dealer stands at 17
build_output <- function(out_deck, runs) {
  count_obs <- numeric(runs)
  vec16 <- numeric(runs)
  vec17 <- numeric(runs)
  vec18 <- numeric(runs)
  vec19 <- numeric(runs)
  vec20 <- numeric(runs)
  for (i in 1:runs){
    count_obs[i] <- i
    vec16[i] <- play_deck2(out_deck,16)
    vec17[i] <- play_deck2(out_deck,17)
    vec18[i] <- play_deck2(out_deck,18)
    vec19[i] <- play_deck2(out_deck,19)
    vec20[i] <- play_deck2(out_deck,20)
  }
  output <- data.frame("count_obs"= count_obs,
                       "stand_16"=vec16,"stand_17"=vec17,
                       "stand_18"=vec18,"stand_19"=vec19,
                       "stand_20"=vec20)
```

```
  return(output)
}
```

3.5 Run 12 Simulations

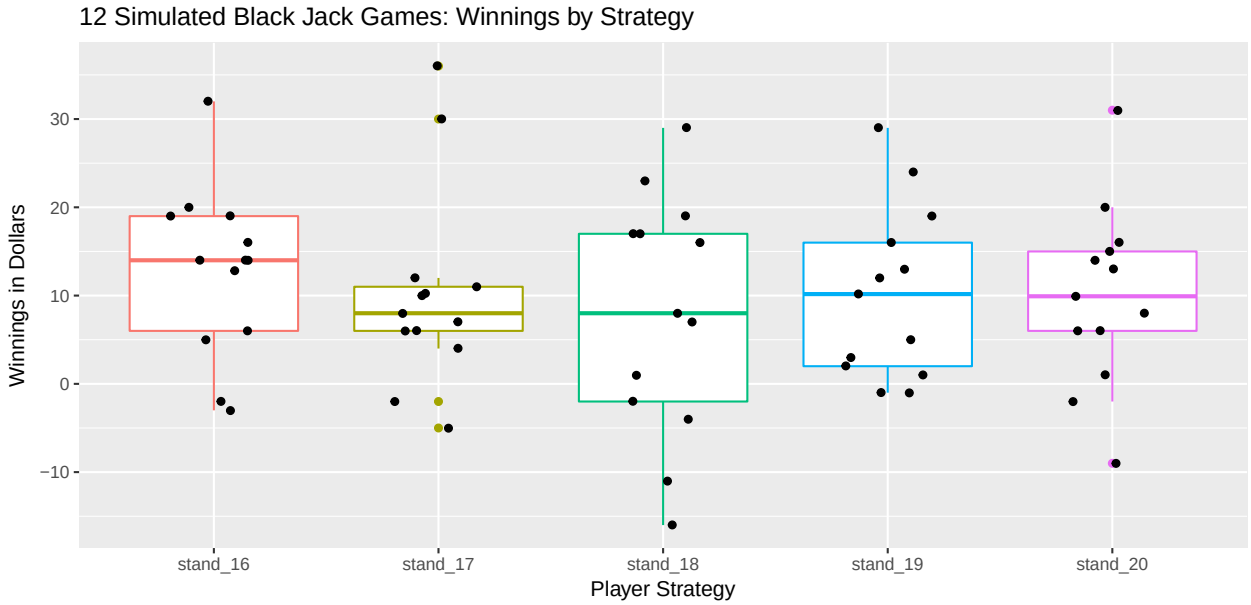
```
set.seed(2345)
# Create set of two decks, simulate 12 two-deck games; run sim with diff strategies

# vector for two decks
two_deck <- c(2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,7,7,7,7,8,8,8,8,
              9,9,9,9,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
              11,11,11,11,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,
              7,7,7,7,8,8,8,8,9,9,9,9,10,10,10,10,10,10,10,10,10,10,
              10,10,10,10,10,11,11,11,11)

# 12 games per strategy
output12 <- build_output(two_deck,12)
output12_avg <- c("stand_16"=mean(output12$stand_16),
                  "stand_17"=mean(output12$stand_17),
                  "stand_18"=mean(output12$stand_18),
                  "stand_19"=mean(output12$stand_19),
                  "stand_20"=mean(output12$stand_20))
```

Table 1: Winnings by Strategy

stand_16	12.83
stand_17	10.25
stand_18	8.00
stand_19	10.17
stand_20	9.92

If you were to only look at the average winnings across 12 games, it would appear that standing at 16 is the best option when the dealer stands at 17. The below plot though illustrates just how much variation there is in results. Simulations of 100, 500, 1000, and 1,500 rounds are made below to improve decision-making.

3.6 Plot the results



3.7 Increasing the Number of Simulations

```
set.seed(2345)
# 100 simulations
output100 <- build_output(two_deck,100)

# 500 simulations
output500 <- build_output(two_deck,500)

# 1000 simulations
output1000 <- build_output(two_deck,1000)

# 1500 simulations
output1500 <- build_output(two_deck,1500)
```

3.7.1 Clean up the data for plotting

```
plot_df <- rbind(output100_avg,
            output500_avg,
            output1000_avg,
            output1500_avg)
#remove row names
row.names(plot_df) <- NULL
#add column to record sim size
plot_df <- cbind(Simulation_Size=c(100,500,1000, 1500),plot_df)
#convert matrix to df
plot_df <- data.frame(plot_df)
plot_df2 <- round(plot_df,2)
```

```
#convert data to long format
plot_df <- melt(plot_df, id.vars="Simulation_Size",
                value.name="value",
                variable.name="Strategy")
```

3.8 Plot updated results

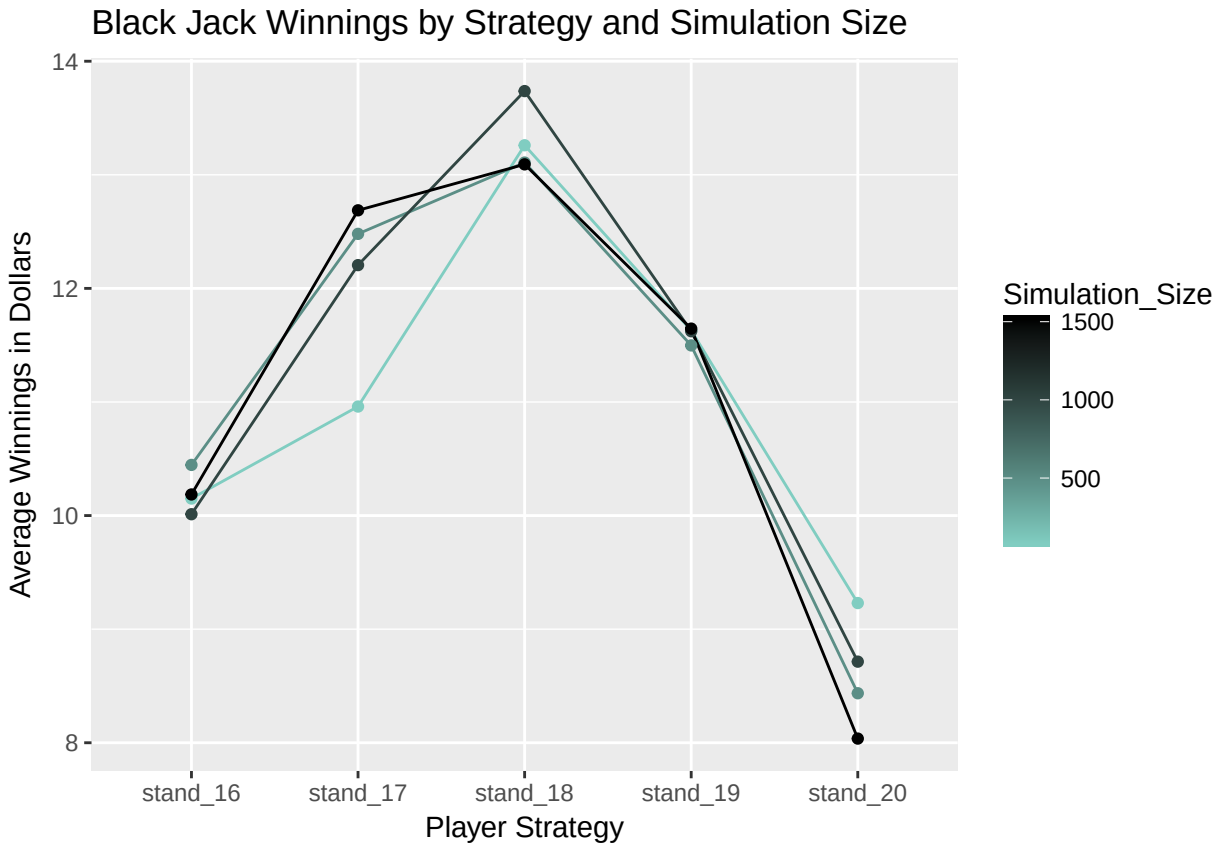


Table 2: Average Winnings by Strategy and Simulation Size

Simulation_Size	stand_16	stand_17	stand_18	stand_19	stand_20
100	10.15	10.96	13.26	11.63	9.23
500	10.45	12.48	13.11	11.50	8.44
1000	10.01	12.21	13.74	11.62	8.71
1500	10.19	12.69	13.09	11.65	8.04

As the number of simulations increases, the results converge. It appears to be that standing at 18 is the best option, as standing at 19 leaves the player with fewer winnings. Similarly, standing at 16, 17, or 20 results in fewer winnings overall. Therefore if the dealer stands at 17, the player should stand at 18.

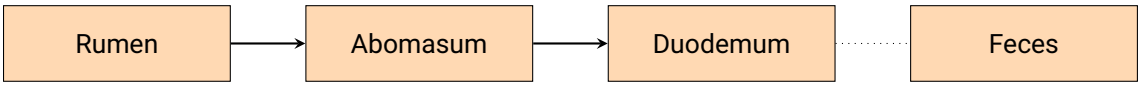
4 Textbook Part III:

4.1 The problem

The digestive processes of sheep can highlight the nutritionally value in varied feeding schedules or varied food preparation. This is especially important when raising sheep for commercial purposes.

4.2 The digestive process

Sheep are a cud-chewing animal which means that unchewed food goes through a series of storage stomachs called the rumen and the reticulum. The process is illustrated below:



4.3 The experiment

The digestive process is most observable at the beginning and at the end, we can observer and control what goes in and what comes out.

4.4 The model

Suppose that at $t = 0$ a sheep is fed an amount R of food which goes immediately into its rumen. This food will pass gradually from the rumen through the abomasum into the duodenum. At any later time t we shall define:

$r(t)$ = the amount of food still in the rumen; $a(t)$ = the amount in the abomasum; $d(t)$ = the amount which by then has arrived in the duodenum.

So $r(0) = R$, $a(0)$ = $d(0)$ = 0, and, for all t>0,

(1) $r(t) + a(t) + d(t) = R.$

4.5 The assumptions

Two assumptions are made:

(A) Food moves out of the rumen at a rate proportional to the amount of food in the rumen. Mathematically this says:

(2) $r'(t) = -k_1 r(t)$

where k_1 is a positive proportionality constant.

(B) Food moves out of the abomasum at a rate proportional to the amount of food in the abomasum. Since at the same time food is moving into the abomasum at the rate given by Equation (2), the assumption says

(3) $a'(t) = k_1 r(t) - k_2 a(t)$

where k_2 is another positive proportionality constant.

4.6 The solutions of the equations

4.6.1 Solving for $r(t)$

It is straightforward to solve Equation (2) for $r(t)$.
We just divide through by $r(t)$ and then integrate from 0 to t:

$$\int_0^t \frac{r'(t)}{r(t)}dt = -\int_0^t k_1 dt$$

$$ln(\frac{r(t)}{R}) = -k_1 t$$

,

since r(0) = R, and finally:

(4) $r(t) = Re^{-k_1 t}$

4.6.2 Solving for $a(t)$

Finding $a(t)$ is a bit more tricky. Applying Equations (4) to Equation (3) we get:

(5) $a'(t) = k_1 Re^{-k_1 t} - k_2 a(t)$

Equation (5) probably looks quite different from any you have seen before. Let us try to make a shrew guess what kind of solution it has. It says that the derivative of $a(t)$ is the sum of two terms, $k_1 Re^{-k_1 t}$ and $-k_2 a(t)$. With luck, this might remind us of the product rule:

(6) if $a(t) = u(t) \bullet v(t)$ then $a'(t) = u(t) \bullet v'(t) + v(t) \bullet u'(t)$

Can we pick $u(t)$ and $v(t)$ so the terms in Equation (6) match up with the terms in Equation (5)? In other words, can we pick $u(t)$ and $v(t)$ so that

(7) $u(t) \bullet v'(t) = k_1 Re^{-k_1^t}$

and

(8) $v(t) \bullet u'(t) = -k_2 a(t)$

Since $a(t) = u(t) \bullet v(t)$, Equation (8) can rewritten $v(t) \bullet u'(t) = -k_2 u(t) v(t)$, we are in business! The $v(t)$ factors cancel out, leaving us with

$$u'(t) = -k_2 u(t)$$

which looks very much like Equation (2) and can be solved in the same way.

$\int_0^t \frac{u'(t)}{u(t)}dt = -\int_0^t k_2 dt$

Writing K = u(0):

$ln(\frac{u(t)}{K}) = -k_2 t$
$u(t) = Ke^{-k_2 t}$.

Putting this into Equation (7) gives

$Ke^{-k_2 t} v'(t) = k_1 Re^{-k_1 t}$
$v'(t) = \frac{k_1 R}{K} e(k_2 - k_1)^t$.

If $k_1 = k_2$ we feel confident you can complete this solution yourself (Exercise 1).

4.7 Exercise 1. Find $a(t)$ if $k_1 = k_2$.

$$a'(t) = u(t) \bullet v'(t) + v(t) \bullet u'(t)$$

Using the solution from equation 8 we get:

$$u(t) = Ke^{-k_2 t}$$

Using the solution from equation 7 we get:

$$v'(t) = \frac{k_1 R}{K} e^{k_2 - k_1} t$$

Substituting the solutions for $a(t)$ we get:

$$\frac{k_1 R}{K} e^{k_2 - k_1} t * Ke^{-k_2 t}$$

If k_1 = k_2 then $e^0 = 1$

$$\frac{k_1 R}{K} t * Ke^{-k_2 t}$$

K factors out :

$$k_1 R t e^{-k_2 t}$$

Solution :

$$a(t) = k_1 R t e^{-k_2 t}$$

If $k_1 \neq k_2$, then $k_2 - k_1 \neq 0$ and so we can write

$$v(t) = \frac{k_1 R}{K(k_2 - k_1)} e^(k_2 - k_1)^t + c$$

Where C is the constant of integration. Then

$$a(t) = u(t) \bullet v(t) = \frac{k_1 R}{k_2 - k_1} e^{-k_1 t} + CKe^{-k_2 t}$$

Using the fact that $a(0) = 0$, we get

$$0 = \frac{k_1 R}{k_2 - k_1} + CK$$

$$CK = -\frac{k_1 R}{k_2 - k_1}$$

(9) $a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$

4.8 Exercise 2.

4.8.1 (a) Find the time t at which $a(t)$ is maximum.

$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$

$$0 = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$

Divide both sides by $\frac{k_1 r}{k_2 - k_1}$

$$e^{-k_2 t} k2 - e^{-k_1 t} k1 = 0$$

Factor $e^{-k_1 t}$, $e^{-k_2 t}$:

$$-e^{-k_1 t - k_2 t}(e^{k_2 t} k_1 - e^{k_1 t} k2) = 0$$

Multiply by -1 to simplify and split into two equations:

$$e^{-k_1 t - k_2 t} = 0 \; and \; e^{k_2 t} k_1 - e^{k_1 t} k2 = 0$$

Since $e^(z)$ can never be zero for any real number, no solution exists for

$$e^{-k_1 t - k_2 t} \neq 0$$

Divide both sides by e^{k2t}:

$$k_1 - e^{t(k_1 - k_2)} k2 = 0$$

Subtract k1:

$$-e^{t(k_1 - k_2)} k2 = -k_1$$

Divide both sides by -k2:

$$e^{t(k_1 - k_2)} = \frac{k_1}{k_2}$$

Take the log:

$$t(k_1 - k_2) = \log(\frac{k_1}{k_2})$$

Solve for t:

$$t = \frac{\log(\frac{k_1}{k_2})}{(k_1 - k_2)}$$

Alternate Solution:

$$t = \frac{lnk_1 - lnk_2}{k_1 - k_2}$$

4.8.2 (b) Find the maximum value of $a(t)$.

$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$

Substituting t we get :

$$\frac{k1R}{k2 - k1}(e^{-k1\frac{\log(\frac{k1}{k2})}{(k1-k2)}} - e^{-k2\frac{\log(\frac{k1}{k2})}{(k1-k2)}})$$

Using the chain rule we identify that:

$$e^{-k1\frac{\log(\frac{k1}{k2})}{(k1-k2)}} = \frac{k_1}{k_2}^{-\frac{k_1}{k_1-k_2}}$$

Therefore with substitution and applying the chain rule the maximum value of $a(t)$ is :

$$\frac{k_1 R}{k_2 - k_1}[(\frac{k_1}{k_2})^{-\frac{k_1}{k_1-k_2}} - (\frac{k_1}{k_2})^{-\frac{k_2}{k_1-k_2}}]$$

4.9 Exercise 3.

If k_1 = 2 and k_2 = 1, how much food must the abomasum be able to hold if a meal of amount R is fed at time t = 0? If we substitute our values we have:

$$\frac{2R}{2-1}[(\frac{2}{1})^{-\frac{1}{2-1}} - (\frac{2}{1})^{-\frac{2}{2-1}}]$$

Simplify:

$$2[(\frac{2}{1})^{-1} - (\frac{2}{1})^{-2}]R$$

$$2[(\frac{1}{2}) - (\frac{1}{4})]R$$

$$2[(\frac{2}{4}) - (\frac{1}{4})]R$$

$$2[\frac{1}{4}]R$$

answer:

$$\frac{1}{2}R$$

4.10 Comparison of the Model's Predictions with Experimental Data

Equations (4) and (9) are purely theoretical and are based on assumptions. To confirm their accuracy we will see if they agree with experimental data. Since the data concern is fecal excretion as a function of time, we must first translate our results into results on fecal excretion.

4.11 Solving for $d(t)$

Starting with Equation (1), and using Equations (4) and (9) (still assuming $k_1 \neq k_2$):

$$d(t) = R - r(t) - a(t)$$

$$= R - Re^{-k_1 t} - \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e{-k_2 t})$$

$$= R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 t} - k_1 e^{-k_2 t})$$

4.12 The Formula for the Amount of Feces

Recall that $d(t)$ is the total amount of food which has entered the duodenum by time t, including food already excreted. Since excretion is not a continuous process, we cannot hope to represent it by an equation involving derivatives. Instead, we simply note that all food arriving in the duodenum is excreted after a certain time delay. Let us suppose that the average time delay is T hours. In other words, the amount of feces produced by any time t > T is, on the average, the amount of food which had already entered the duodenum T hours earlier, at time t - T. If f(t) denotes the amount of feces produced by time t, this says:

$f(t) = d(t - T) for all t > T$

(10) $f(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1(t-T)} - k_1 e^{-k_2(t-T)})$

for all t > T.

4.13 Exercise 4.

Remembering that r(t) is the total amount of food which has entered the the rumen by time t. There is a continuous flow from the rumen to the abomasum

4.13.1 (a) Determine the values of t for which r'(t) is, respectively, positive, negative, and zero.

$$r(t) = Re^{-k_1 t}$$

Factor out constants:

$$= R(\frac{d}{dt}(e^{-k_1 t}))$$

Using the chain rule :

$$= e^{-tk_1}(\frac{d}{dt}(-tk_1))R$$

factor out constant:

$$= -k_1(\frac{d}{dt}(t))e^{-k_1 t}R$$

derivative of t is 1:

$$r'(t) = -e^{-tk_1}k_1R$$

$r'(t) = -e^{-tk_1}$ and therefore $r'(t) < 0$ for all t.

4.13.2 (b) Determine the values of t for which r"(t) is, respectively, positive, negative, and zero.

$$= \frac{d}{dt}(R(\frac{d}{dt}(e^{-tk_1})))$$

Using the chain rule:

$$= \frac{d}{dt}(e^{-tk_1}(\frac{d}{dt}(-tk_1))R)$$

factor out constants:

$$= \frac{d}{dt}(R - (\frac{d}{dt}(t))k_1e^{-tk_1})$$

derivative of t is 1:

$$= \frac{d}{dt}(R(e^{-tk_1} - k_1))$$

factor out constants:

$$= R(\frac{d}{dt}(e^{-tk_1}))k_1$$

using the chain rule:

$$= -Rk_1e^{-tk_1}(\frac{d}{dt}(-tk_1))$$

factor out constants and simplify

$$= -e^{-tk_1}R - (\frac{d}{dt}(t))k_1^2$$

derivative of t is 1:

$$r''(t) = e^{-tk_1}Rk_1^2$$

$r''(t) = e^{-tk_1}$ therefore for all real solutions : $r''(t) > 0$ for all t

4.13.3 (c) Determine $\lim_{t\to\infty} r(t)$

$$r(t) = Re^{-k_1t}$$
$$r(t) = Re^{-k_1\infty}$$

Since $e^{-\infty}$ = 0 our solution is: $\lim_{t\to\infty} r(t) = 0$

4.14 Exercise 5.

Note that for exercise 5, 6, and 7 $t_0 = \frac{lnk_1 - lnk_2}{k_1 - k_2}$

Remembering that a(t) is the total amount of food which has entered the the abomasum by time t from the rumen and exiting to the duodenum. There is a continuous flow from the abomasum to the duodenum.

We have previously solved for a(t) and $a'(t)$

$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$

$$a'(t) = k_1 R e^{-k_1 t} - k_2 a(t)$$

$$= \frac{d}{dt}(\frac{(e^{-k_1 t} - e^{-k_2 t})k_1 R}{-k_1 + k_2})$$

Factor out constant:

$$= \frac{k_1 R(\frac{d}{dt}(e^{-k_1 t}) - (e^{-k_2 t}))}{-k_1 + k_2}$$

Differentiate the sum term by term:

$$= \frac{\frac{d}{dt}(e^{-k_1 t}) - \frac{d}{dt}(e^{-k_2 t})k_1 R}{-k_1 + k_2}$$

Use the chain rule:

$$\frac{k_1 R(-(\frac{d}{dt}(e^{-k_2 t})) + \frac{\frac{d}{dt} - k_1 t}{e^{k_1 t}}}{-k_1 + k_2}$$

Factor out constants:

$$\frac{k_1 R(-(\frac{d}{dt}(e^{-k_2 t})) + \frac{-k_1 \frac{d}{dt} t}{e^{k_1 t}}}{-k_1 + k_2}$$

The derivative of t is 1 and using the chain rule:

$$\frac{k_1 R(-\frac{k_1}{e^{k_1 t}} - \frac{\frac{d}{dt} - k_2 t}{e^{k_1 t}}}{-k_1 + k_2}$$

Factor out constants and the derivative of t is 1:

$$= \frac{k_1(-\frac{k_1}{e^{k_1 t}} + \frac{k_2}{e^{k_2 t}})R}{-k_1 + k_2}$$

$$a'(t) = \frac{k_1(-\frac{k_1}{e^{k_1 t}} + \frac{k_2}{e^{k_2 t}})R}{-k_1 + k_2}$$

4.14.1 (a) Determine the values of t for which $a'(t)$ is, respectively, positive, negative, and zero.

$$a'(t) = \frac{k_1(-\frac{k_1}{e^{k_1 t}} + \frac{k_2}{e^{k_2 t}})R}{-k_1 + k_2}$$

$$a'(t) = \frac{k_1(-\frac{k_1}{e^{k_1 t}} + \frac{k_2}{e^{k_2 t}})R}{-k_1 + k_2}$$

$$= \frac{k_1(-\frac{k_1}{e^{k_1 t_1}} + \frac{k_2}{e^{k_2 t_1}})R}{-k_1 + k_2}$$

Therefore assuming that:

$k_1 > 0$ and $k_2 > k_1$ and $t_0 = \frac{log(\frac{k_1}{k_2})}{k_1 - k_2}$; $a'(t)$ is positive or $a'(t) > 0$ for all $t < t_0$;

$$= \frac{k_1(-\frac{k_1}{e^{k_1 * 0}} + \frac{k_2}{e^{k_2 * 0}})R}{-k_1 + k_2}$$

$t_0 = \frac{lnk_1 - lnk_2}{k_1 - lnk_2}$

$$= \frac{k_1(-\frac{k_1}{e^{k_1 * \frac{lnk_1 - lnk_2}{k_1 - lnk_2}}} + \frac{k_2}{e^{k_2 * \frac{lnk_1 - lnk_2}{k_1 - lnk_2}}})R}{-k_1 + k_2}$$

$$\frac{k_1 R(k_1 \frac{k_1}{k_2}^{\frac{k_1}{log(k_2) - k_1}} - k_2 \frac{k_2}{k_1}^{\frac{k_2}{k_1 - log(k_2)}})}{k_1 - k_2}$$

Therefore : $a'(t_0) = 0$;

We know from our previous exercise we can conclude that $a'(t)$ is negative or $a'(t) < 0$ for $t > t_0$.

4.14.2 (b) Determine the values of t for which a"(t) is, respectively, positive, negative, and zero.

$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$

Differentiate the sum, factor constants, and use the chain rule :

$$\frac{d}{dt}(\frac{k_1 R(e^{-k_1 t}\frac{d}{dt}(-k_1 t) - \frac{d}{dt}(e^{-k_2 t}))}{-k_1 + k_2})$$

Factor out constants, derivative of t is 1, and use the chain rule :

$$\frac{d}{dt}(\frac{k_1 R(e^{-k_1 t}(-k_1) - e^{-k_2 t}\frac{d}{dt}(-k_2 t))}{-k_1 + k_2})$$

Factor out constants, derivate of t is 1, and differentiate the sum term:

$$= k_2 \frac{d}{dt}(e^{-k_2 t}) - k_1 \frac{d}{dt}(e^{-k_1 t})\frac{k_1 R}{-k_1 + k_2}$$

Using the chain rule, factor out constants, derivative of t is 1:

$$= \frac{k_1 R(k_2 \frac{d}{dt}(e^{-k_2 t})) + e^{-k_1 t}k_1^2)}{-k_1 + k_2}$$

Using the chain rule, factor out constants, and the derivative of t is 1:

$$a''(t) = \frac{k_1(e^{-k_1 t}k_1^2 - e^{-k_2 t}k_2^2)R}{-k_1 + k_2}$$

Note that $t_0 = \frac{lnk_1 - lnk_2}{k_1 - k_2}$

$$\frac{k_1(e^{-k_1 \frac{log(k1) - log(k2)}{k1 - k2}}k_1^2 - e^{-k_2 \frac{log(k1) - log(k2)}{k1 - k2}}k_2^2)R}{-k_1 + k_2}$$

$$\frac{k_1 R(k_1^2(\frac{k_1}{k_2}^{\frac{k_1}{k_1-k_2}} - k_2^2(\frac{k_1}{k_2})^{-\frac{k_2}{k_1-k_2}}))}{k_2 - k_1}$$
$$-k_1 k_2 R(\frac{k_1}{k_2}^{-\frac{k_1}{k_1-k_2}})$$

Therefore we can see that $a''(t) < 0$ for all $t < 2t_0$;

$$\frac{k_1(e^{-k_1 2\frac{log(k1)-log(k2)}{k1-k2}}k_1^2 - e^{-k_2 2\frac{log(k1)-log(k2)}{k1-k2}}k_2^2)R}{-k_1 + k_2}$$

Assuming k_1, k_2, and R are real numbers then $a''(2t_0) = 0$;

As we saw from the previous exercise that $a''(t) > 0$ for $t > 2t_0$.

4.14.3 (c) Determine $\lim_{t\to\infty} a(t)$

$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 t} - e^{-k_2 t})$$
$$a(t) = \frac{k_1 R}{k_2 - k_1}(e^{-k_1 \infty} - e^{-k_2 \infty})$$

Again, as we saw previously $e^{-\infty}$ = 0 so our solution : $\lim_{t\to\infty} a(t) = 0$

4.15 Exercise 6.

$$d(t) = R - r(t) - a(t)$$
$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 t} - k_1 e^{-k_2 t})$$

4.15.1 (a) Determine the values of t for which d'(t) is, respectively, positive, negative, and zero.

$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 t} - k_1 e^{-k_2 t})$$
$$-\frac{R(\frac{d}{dt}(-e^{-k_2 t}k1 + e^{-k1t}k2))}{-k_1 + k_2} + \frac{d}{dt}(R)$$

$$\frac{d}{dt}(R) - k_2\frac{d}{dt}(e^{-k_1 t}) - k1\frac{d}{dt}(e^{-k_2 t})\frac{R}{-k_1 + k_2}$$

Using the chain rule and derivative of R is zero:

$$-\frac{R(-k_1\frac{d}{dt}(e^{-k_2 t})) + e^{-k_1 t}\frac{d}{dt}(-k_1 t)k_2)}{-k_1 + k_2}$$

Factor out constants and derivative of t is 1:

$$-\frac{R(-k_1\frac{d}{dt}(e^{-k_2 t})) + 1e^{-k_1 t}k_2)}{-k_1 + k_2}$$

Using chain rule:

$$-\frac{R(-e^{-k_1 t}k_1 k_2 - e^{-k_2 t}\frac{d}{dt}(-k_2 t)k_1}{-k_1 + k_2}$$

Factor out:

$$-\frac{R(-e^{-k_1 t}k_1k_2 - -k_2\frac{d}{dt}(t)e^{-k_2 t}k_1)}{-k_1+k_2}$$

Simplify and derivative of t is 1:

$$d'(t) = -\frac{(-e^{-k_1 t}k_1k_2 + e^{-k_2 t}k_1k_2)R}{-k_1+k_2}$$

Note $t_0 = \frac{lnk_1 - lnk_2}{k_1-k_2}$

$$-\frac{(-e^{-k_1\frac{lnk_1-lnk_2}{k_1-k_2}}k_1k_2 + e^{-k_2\frac{lnk_1-lnk_2}{k_1-k_2}}k_1k_2)R}{-k_1+k_2}$$

$$\frac{k_1k_2R(\frac{k_2}{k_1}^{\frac{k_2}{k_1-k_2}} - \frac{k_2}{k_1}^{\frac{k_1}{k_1-k_2}})}{k_1-k_2}$$

$$k_2R(\frac{k_2}{k_1})^{\frac{k_2}{k_1-k_2}}$$

Therefore we can conclude that $d'(t)$ is positive or $d'(t) > 0$ for all t (must have $k_1 > k_2$).

4.15.2 (b) Determine the values of t for which d"(t) is, respectively, positive, negative, and zero.

$$d'(t) = -\frac{(-e^{-k_1 t}k_1k_2 + e^{-k_2 t}k_1k_2)R}{-k_1+k_2}$$

$$= -\frac{R(k_1k_2(\frac{d}{dt}(e^{-k_2 t})) - -k_1\frac{d}{dt}(t)e^{-k_1 t}k_1k_2)}{-k_1+k_2}$$

Simplify and the derivative of t is 1:

$$= -\frac{R(k_1k_2(\frac{d}{dt}(e^{-k_2 t})) - e^{-k_1 t}k_1^2k_2)}{-k_1+k_2}$$

Use the chain rule:

$$= -\frac{R(e^{-k_1 t}k_1^2k_2 + e^{-k_2 t}\frac{d}{dt}(t)k_1k_2}{-k_1+k_2}$$

Factor out constants:

$$= -\frac{R(e^{-k_1 t}k_1^2k_2 + -k_2\frac{d}{dt}(t)e^{-k_2 t}k_1k_2}{-k_1+k_2}$$

Derivative of t is 1 and simplify:

$$= -\frac{R(e^{-k_1 t}k_1^2k_2) - e^{k_2 t}k_1k_2^2)}{-k_1+k_2}$$

$$d''(t) = \frac{k_1k_2Re^{t(-k_1-k_2)}(k_1e^{k_2 t} - k_2e^{k_1 t})}{k_1-k_2}$$

$$(k_1-k_2)(k_1e^{k_2 t} - k_2e^{k_1 t}) > 0$$

Therefore $d''(t)$ is positive or $d''(t) > 0$ for all $t < t_0$;

Note that $t_0 = \frac{lnk_1 - lnk_2}{k_1 - k_2}$

$$\frac{k_1 k_2 R e^{t(-k_1-k_2)}(k_1 e^{k_2 \frac{lnk_1-lnk_2}{k_1-k_2}} - k_2 e^{k_1 \frac{lnk_1-lnk_2}{k_1-k_2}})}{k_1 - k_2}$$

$$\frac{k_1 k_2 R(k_1(\frac{k_1}{k_2})^{\frac{k_2}{k_1-k_2}} - k_2(\frac{k_1}{k_2}^{\frac{k_1}{k_1-k_2}}))e^{t(-k_1-k_2)}}{k_1 - k_2}$$

$$\frac{k_1^2 k_2 R(\frac{k_1}{k_2})^{\frac{k_1}{k_1-k_2}} e^{\frac{(-k_1-k_2)(log(k_1)-log(k_2))}{k_1-k_2}}}{k_1 - k_2} - \frac{k_1 k_2^2 R(\frac{k_1}{k_2})^{\frac{k_1}{k_1-k_2}} e^{\frac{(-k_1-k_2)(log(k_1)-log(k_2))}{k_1-k_2}}}{k_1 - k_2}$$

Therefore $d''(t)$ or $d''(t_0) = 0$;

If the first part of the exercise is indeed true then $d''(t)$ is negative or $d''(t) < 0$ for $t > t_0$.

4.15.3 (c) Determine $\lim_{t\to\infty} d(t)$

$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 t} - k_1 e^{-k_2 t})$$

$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 \infty} - k_1 e^{-k_2 \infty})$$

$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1 \infty} - k_1 e^{-k_2 \infty})$$

Since $e^{-\infty}$ = 0 :

$$d(t) = R - \frac{R}{k_2 - k_1}(k_2 * 0 - k_1 * 0)$$

Therefore, our solution is : $\lim_{t\to\infty} d(t) = R$

4.16 Exercise 7.

For the function f(t) follow the instructions of Exercise 4.

$$f(t) = R - \frac{R}{k_2 - k_1}(k_2 e^{-k_1(t-T)} - k_1 e^{-k_2(t-T)})$$

4.16.1 (a) Determine the values of t for which f'(t) is, respectively, positive, negative, and zero.

$$= -\frac{R\frac{d}{dt}(-e^{-k_2(t-T)}k_1 + e^{-k_1(t-T)}k_2))}{-k_1 + k_2} + \frac{d}{dt}(R)$$

Factor out constants:

$$= \frac{d}{dt}(R) - k_2(\frac{d}{dt}(e^{-k_1(t-T)})) - k_1(\frac{d}{dt}(e^{-k_2(t-T)}))\frac{R}{-k_1 + k_2}$$

Using the chain rule:

$$= \frac{d}{dt}(R) - \frac{R(-k_1)(\frac{d}{dt}(e^{-k_2(t-T)})) + e^{-k_1(t-T)}\frac{d}{dt}(-k_1(t-T)))k_2)}{-k_1+k_2}$$

Factor out constants:

$$= \frac{d}{dt}(R) - \frac{R(-k_1(\frac{d}{dt}(e^{-k_2(t-T)}))) + -k_1(\frac{d}{d_t}(t-T))e^(-k_1(t-T))k_2)}{-k_1+k_2}$$

Differentiate the sum term:

$$= \frac{d}{dt}(R) - \frac{R(-k_1(\frac{d}{dt}(e^{-k_2(t-T)}))) + \frac{d}{dt}(t) + \frac{d}{dt}(-T)e^{-k1(t-T)k_1k_2}}{-k_1+k_2}$$

Derivative of t is 1 and R is 0:

$$= -\frac{R(-k_1(\frac{d}{dt}(e^{-k_2(t-T)}))) + \frac{d}{dt}(-T)e^{-k1(t-T)k_1k_2}}{-k_1+k_2}$$

Using the chain rule:

$$= -\frac{R(-e^{-k_1(t-T)}k_1k_2(\frac{d}{dt}(-T))) - e^{-k_2(t-T)}(\frac{d}{dt}(-k_2(t-T)))k_1}{-k_1+k_2}$$

Factor out constants:

$$= -\frac{R(-e^{-k_1(t-T)}k_1k_2(\frac{d}{dt}(-T))) - -k_2(\frac{d}{dt}(t-T))e^{-k_2(t-T)}k_1}{-k_1+k_2}$$

Differentiate the sum term:

$$= -\frac{R(-e^{-k_1(t-T)}k_1k_2(\frac{d}{dt}(-T)) + \frac{d}{dt}(t) + \frac{d}{dt}(-T)e^{-k_2(t-T)}k_1k_2)}{-k_1+k_2}$$

The derivative of t is 1 and -T is zero

$$= -\frac{R(-e^{-k_1(t-T)}k_1k_2 + e^{-k_2(t-T)}k_1k_2)}{-k_1+k_2}$$

$$f'(t) = -\frac{-e^{-k_1(t-T)}k_1k_2 + e^{-k_2(t-T)}k_1k_2)R}{-k_1+k_2}$$

note $t_0 = \frac{lnk_1 - lnk_2}{k_1-k_2}$.

$$(k_1-k_2)(e^{k_1(T-t)} - e^{k_2(T-t)}) > 0$$

Therefore $f'(t)$ is positive or $f'(t) > 0$ for all $t > T$.

4.16.2 (b) Determine the values of t for which f"(t) is, respectively, positive, negative, and zero.

$$= \frac{d}{dt}(\frac{d}{dt}(R - \frac{(-e^{-k_2(t-T)}k_1) + e^{-k_1(t-T)}k_2)R}{-k_1+k_2}))$$

Differentiate and factor out:

$$= \frac{d}{dt}(\frac{d}{dt}(R) - \frac{R\frac{d}{dt}(-e^{-k_2(t-T)}k_1 + e^{-k_1(t-T)}k_2)}{-k_1+k_2}))$$

Factor out:

$$= \frac{d}{dt}(-\frac{R}{-k_1+k_2}k_2\frac{d}{dt}(e^{-k_1(t-T)}) - k_1\frac{d}{dt}(e^{-k_2(t-T)}) + \frac{d}{dt}(R))$$

Using the chain rule:

$$= \frac{d}{dt}(-\frac{R(e^{-k_1(t-T)}\frac{d}{dt}(-k_1(t-T))k_2 - k_1(\frac{d}{dt}(e^{-k_2(t-T)}))}{-k_1+k_2} + \frac{d}{dt}(R))$$

Factor out:

$$= \frac{d}{dt}(-\frac{R(k_2 - k_1\frac{d}{dt}(t-T)e^{-k_1(t-T)} - k_1(\frac{d}{dt}(e^{-k_2(t-T)})))}{-k_1+k_2} + \frac{d}{dt}(R))$$

Differentiate the sum

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - k_1\frac{d}{dt}(t) + \frac{d}{dt}(-T) - k_1(\frac{d}{dt}(e^{-k_2(t-T)})))}{-k_1+k_2} + \frac{d}{dt}(R))$$

The derivative of t is 1 and -T is zero:

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - k_1(1+0) - k_1(\frac{d}{dt}(e^{-k_2(t-T)})))}{-k_1+k_2} + \frac{d}{dt}(R))$$

Use the chain rule:

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - e^{-k_2(t-T)}\frac{d}{dt}(-k_2(t-T))k_1}{-k_1+k_2} + \frac{d}{dt}(R))$$

Factor out:

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - k_1 - k_2\frac{d}{dt}(t-T)e^{-k_2(t-T)})}{-k_1+k_2} + \frac{d}{dt}(R))$$

Differentiate the sum:

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - k_1(e^{-k_2(t-T)} - k_2\frac{d}{dt}(t) + \frac{d}{dt}(-T)}{-k_1+k_2} + \frac{d}{dt}(R))$$

The derivative of t is 1 and, -T and R is zero:

$$= \frac{d}{dt}(\frac{-R(k_2(e^{-k_1(t-T)}) - k_1(e^{-k_2(t-T)} - k_2(1+0)}{-k_1+k_2} + 0)$$

Factor out:

$$= -\frac{R\frac{d}{dt}(-e^{-k_1(t-T)}k_1k_2 + e^{-k_2(t-T)}k_1k_2))}{-k_1+k_2}$$

Use the chain rule:

$$= -\frac{R(k_1k_2(\frac{d}{dt}(e^{-k_2(t-T)})) - e^{-k_1(t-T)}\frac{d}{dt}(-k_1(t-T))k_1k_2)}{-k_1k_2}$$

Factor out:

$$= -\frac{R(k_1k_2\frac{d}{dt}(e^{-k_2(t-T)})) - -k_1\frac{d}{dt}(t-T)e^{-k_1(t-T)}k_1k_2}{-k_1k_2}$$

Simply

$$= -\frac{R(k_1k_2\frac{d}{dt}(e^{-k_2(t-T)})) + \frac{d}{dt}(t) + \frac{d}{dt}(-T)e^{-k_1(t-T)}k_1^2k_2}{-k_1k_2}$$

The derivative of t is 1:

$$= -\frac{R(k_1k_2\frac{d}{dt}(e^{-k_2(t-T)})) + (1 + \frac{d}{dt}(-T))e^{-k_1(t-T)}k_1^2k_2}{-k_1k_2}$$

Using the chain rule:

$$= -\frac{R(e^{-k_1(t-T)}k_1^2k_2(1+\frac{d}{dt}(t-T)) + -k_2(\frac{d}{dt}(t-T)e^{-k_2(t-T)}k_1k_2)}{-k_1+k_2}$$

Factor and differentiate the sum term by term:

$$= -\frac{R(e^{-k_1(t-T)k_1^2k_2}(1+\frac{d}{dt}(-T)) - \frac{d}{dt}(t) + \frac{d}{dt}(-T)e^{-k_2(t-T)}k_1k_2^2)}{-k_1+k_2}$$

Derivative of t is 1 and -T is zero:

$$= -\frac{R(e^{-k_1(t-T)k_1^2k_2}(1+0) - e^{-k_2(t-T)}k_1k_2^2)(1+0)}{-k_1+k_2}$$

Answer:

$$f''(t) = -\frac{(e^{-k_1(t-T)}k_1^2k_2 - e^{-k_2(t-T)}k_1k_2^2)R}{-k_1+k_2}$$

$$\frac{-Re^{k_1T-k_1t}k_1^2k_2 - Re^{k_2T-k_2t}k_1k_2^2}{k_2-k_1}$$

$$\frac{e^{k_2t-k_2t}k_1k_2^2R}{k_2-k_1} - \frac{e^{k_1t-k_1t}k_1^2k_2R}{k_2-k_1}$$

$$0 < (k_1-k_2)(k_1e^{k_1(T-t)} - k_2e^{k_2(T-t)})$$

Therefore $f''(t)$ is positive or $f''(t) > 0$ for all $T < t < T + t_0$;

Note that $t_0 = \frac{lnk_1 - lnk_2}{k_1-k_2}$

$$0 = \frac{k_1e^{k1(\frac{log(\frac{k_2}{k_1})}{k_1-k_2}+T)} + k_2e^{k2(\frac{log(\frac{k_2}{k_1})}{k_1-k_2}+T)}}{k_1-k_2}$$

$$T = \frac{-log(\frac{k_1}{k_2} + log(k_1) - log(k_2)}{k_1-k_2}$$

Therefore $f''(t)$ is zero at $f''(T+t_0) = 0$;

Having found where $f''(t)$ is positive we can then conclude that $f''(t)$ is negative or $f''(t) < 0$ for $t > T + t_0$.

4.16.3 (c) Determine $\lim_{t\to\infty} f(t)$

$$f(t) = R - \frac{R}{k_2-k_1}(k_2e^{-k_1(t-T)} - k_1e^{-k_2(t-T)})$$

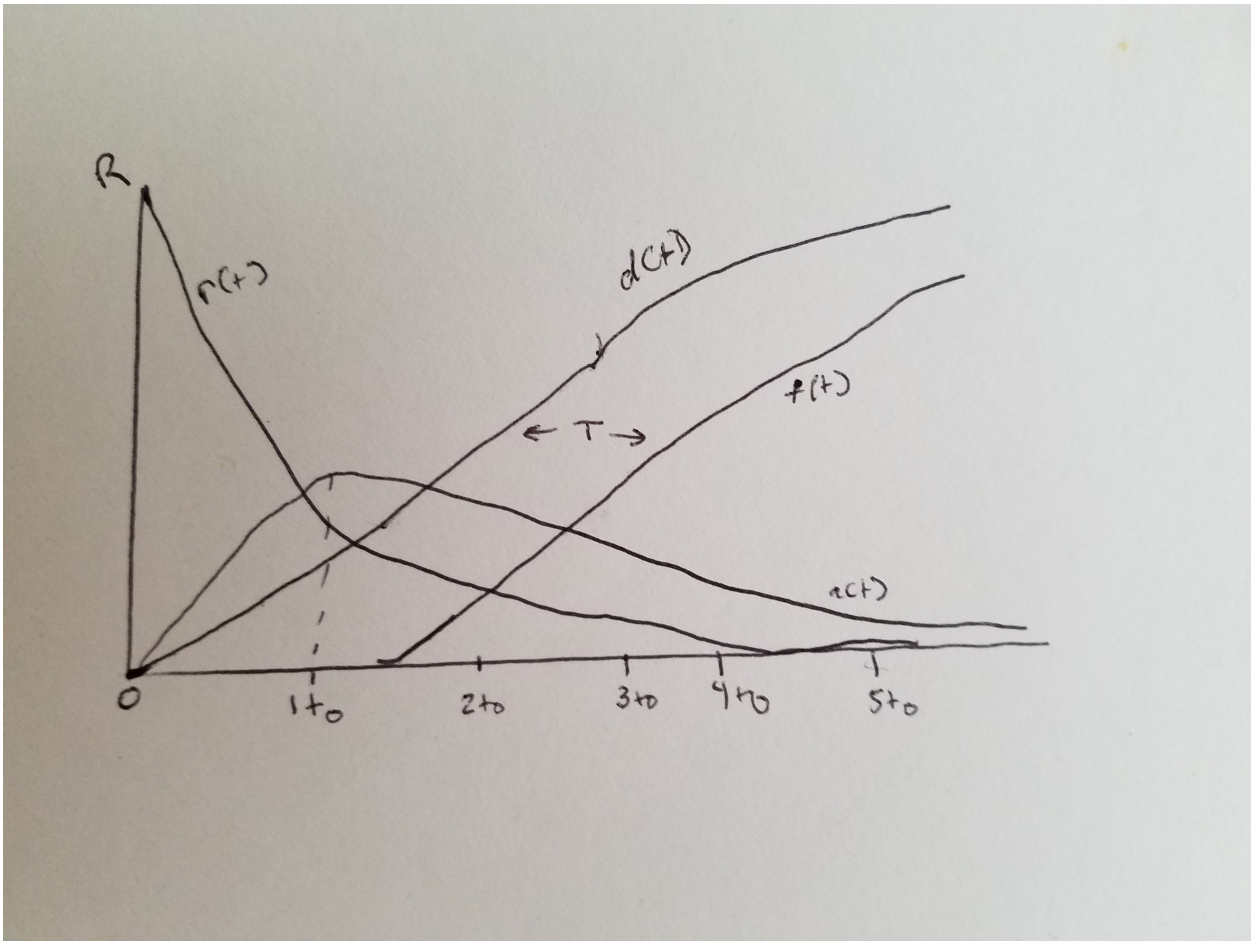
$$f(t) = R - \frac{R}{k_2-k_1}(k_2e^{-k_1(\infty-T)} - k_1e^{-k_2(\infty-T)})$$

Since $e^{-\infty}$ = 0:

$f(t) = R$ so our solution is answer: $\lim_{t\to\infty} f(t) = R$

4.17 Exercise 8.

Using all the information in Exercise 4, 5, 6, and 7 to sketch the graphs of r, a, d, and f. You should not have to compute any points except r(0), a(0), d(0), and f(T), which you already know.



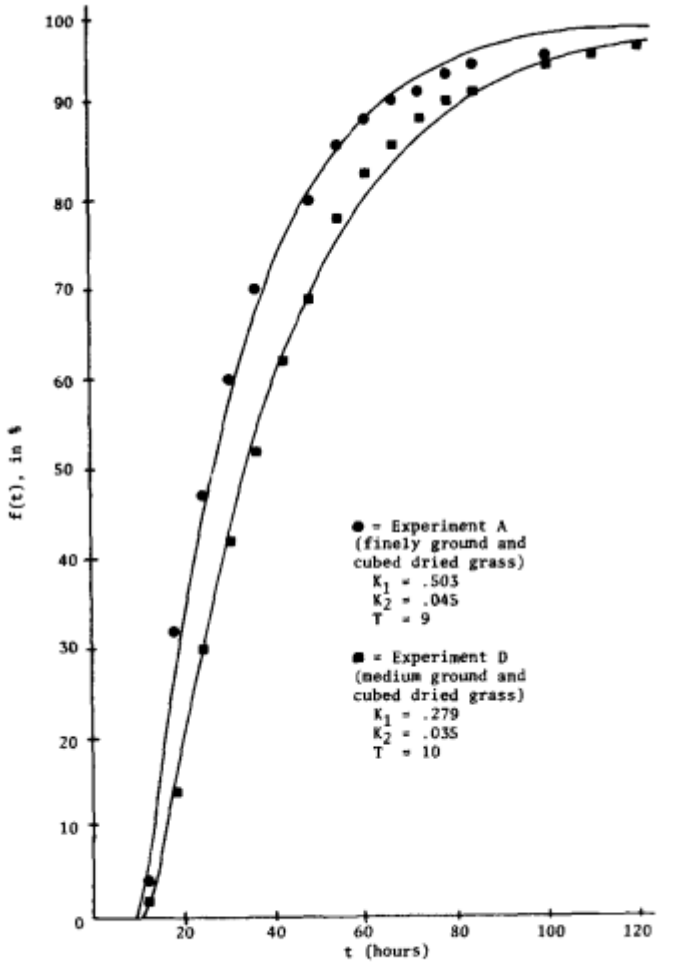


Figure 4. Data from Experiments A and D of Table 1 and Figure 2, with curves fitted from Equation (10). Adapted from Blaxter (1956), p. 76.

Figure 1:

4.18 Exercise 9.

Assuming that increased stay in the digestive system implies improved digestion determine from Figure 4 which method of food preparation is more desirable. Why?

Assuming that we are looking for the feed type that stays in the sheep's stomach the longest, then we would say that the medium-ground grass would be the most desirable. We make this assumption as it stands that the sheep requires less feed for the same results.