Lab 3

Nate Beebe

Due on 03/14/2024 at 11:59 pm

Question 1 In class we computed the run expectancy matrix for the 2016 season. We used this quantity to assess the value of stolen bases and we computed the marginal break even stolen base percentage required to justify an attempt. Do the following:

- Rank catchers by their ability to prevent runs via catching runners attempting to steal bases. Report a top 10 list for most effective catchers at preventing runs via the stolen base. Also report the worse catchers.

```r
library(tidyverse)
library(Lahman)
fields = read_csv("C:/Users/STP/Desktop/stat430/stat430sp25/fields.csv")
dat2016 = read_csv("C:/Users/STP/Desktop/stat430/stat430sp25/all2016.csv",
                    col_names = pull(fields, Header),
                    na = character())
catchers16 = dat2016 %>%
  mutate(RUN1_SB_FL = as.integer(RUN1_SB_FL),
         RUN2_SB_FL = as.integer(RUN2_SB_FL),
         RUN3_SB_FL = as.integer(RUN3_SB_FL),
         RUN1_CS_FL = as.integer(RUN1_CS_FL),
         RUN2_CS_FL = as.integer(RUN2_CS_FL),
         RUN3_CS_FL = as.integer(RUN3_CS_FL))
Names_Catchers = People %>%
  select(nameFirst, nameLast, retroID) %>%
  unite(Name, nameFirst, nameLast, sep = " ")

RE = matrix(c(0.47, 0.25, 0.10,
              0.84, 0.50, 0.22,
              1.06, 0.65, 0.31,
              1.45, 0.94, 0.32,
              1.41, 0.87, 0.42,
              1.75, 1.15, 0.49,
              1.93, 1.34, 0.54,
              2.17, 1.47, 0.76),
            8, 3, byrow=TRUE,
            dimnames =list(c("000","100","010","001",
                             "110","101","011","111"),
                           c("0 outs", "1 out", "2 outs")))


Best_Catchers = catchers16 %>%
  mutate(ST1 = OUTS_CT == 0 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST2 = OUTS_CT == 1 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST3 = OUTS_CT == 2 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST4 = OUTS_CT == 0 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST5 = OUTS_CT == 1 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST6 = OUTS_CT == 2 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID == "",
         ST7 = OUTS_CT == 0 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST8 = OUTS_CT == 1 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST9 = OUTS_CT == 2 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST10 = OUTS_CT == 0 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST11 = OUTS_CT == 1 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST12 = OUTS_CT == 2 & BASE1_RUN_ID == "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST13 = OUTS_CT == 0 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST14 = OUTS_CT == 1 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST15 = OUTS_CT == 2 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID == "",
         ST16 = OUTS_CT == 0 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST17 = OUTS_CT == 1 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST18 = OUTS_CT == 2 & BASE1_RUN_ID != "" & BASE2_RUN_ID == "" & BASE3_RUN_ID != "",
         ST19 = OUTS_CT == 0 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "",
         ST20 = OUTS_CT == 1 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "",
         ST21 = OUTS_CT == 2 & BASE1_RUN_ID == "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "",
         ST22 = OUTS_CT == 0 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "",
         ST23 = OUTS_CT == 1 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "",
         ST24 = OUTS_CT == 2 & BASE1_RUN_ID != "" & BASE2_RUN_ID != "" & BASE3_RUN_ID != "") %>%
  mutate(ST1 = as.integer(ST1),
         ST2 = as.integer(ST2),
         ST3 = as.integer(ST3),
         ST4 = as.integer(ST4),
         ST5 = as.integer(ST5),
         ST6 = as.integer(ST6),
         ST7 = as.integer(ST7),
         ST8 = as.integer(ST8),
         ST9 = as.integer(ST9),
         ST10 = as.integer(ST10),
         ST11 = as.integer(ST11),
         ST12 = as.integer(ST12),
         ST13 = as.integer(ST13),
         ST14 = as.integer(ST14),
         ST15 = as.integer(ST15),
         ST16 = as.integer(ST16),
         ST17 = as.integer(ST17),
         ST18 = as.integer(ST18),
         ST19 = as.integer(ST19),
         ST20 = as.integer(ST20),
         ST21 = as.integer(ST21),
         ST22 = as.integer(ST22),
         ST23 = as.integer(ST23),
         ST24 = as.integer(ST24)) %>%
  mutate("ST4_ST2" = as.integer(ST4 == 1 & RUN1_CS_FL == 1),
         "ST4_ST7" = as.integer(ST4 == 1 & RUN1_SB_FL == 1),
         "ST5_ST3" = as.integer(ST5 == 1 & RUN1_CS_FL == 1),
         "ST5_ST8" = as.integer(ST5 == 1 & RUN1_SB_FL == 1),
         "ST6_END" = as.integer(ST6 == 1 & RUN1_CS_FL == 1),
         "ST6_ST9" = as.integer(ST6 == 1 & RUN2_SB_FL == 1),
         "ST7_ST2" = as.integer(ST7 == 1 & RUN2_CS_FL == 1),
         "ST7_ST10" = as.integer(ST7 == 1 & RUN2_SB_FL == 1),
         "ST8_ST3" = as.integer(ST7 == 1 & RUN2_CS_FL == 1),
         "ST8_ST11" = as.integer(ST7 == 1 & RUN2_SB_FL == 1),
         "ST9_END" = as.integer(ST7 == 1 & RUN2_CS_FL == 1),
         "ST9_ST12" = as.integer(ST7 == 1 & RUN2_SB_FL == 1),
         "ST13_ST5" = as.integer(ST13 == 1 & RUN2_CS_FL == 1 & RUN1_DEST_ID == 1),
         "ST13_ST8" = as.integer(ST13 == 1 & RUN2_CS_FL == 1 & RUN1_DEST_ID == 2),
         "ST13_ST11" = as.integer(ST13 == 1 & RUN1_CS_FL == 1 & RUN2_DEST_ID == 3),
         "ST13_ST19" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 2),
         "ST13_ST16" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 1),
         "ST14_ST6" = as.integer(ST13 == 1 & RUN2_CS_FL == 1 & RUN1_DEST_ID == 1),
         "ST14_ST9" = as.integer(ST13 == 1 & RUN2_CS_FL == 1 & RUN1_DEST_ID == 2),
         "ST14_ST12" = as.integer(ST13 == 1 & RUN1_CS_FL == 1 & RUN2_DEST_ID == 3),
         "ST14_ST20" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 2),
         "ST14_ST17" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 1),
         "ST15_END" = as.integer(ST13 == 1 & RUN2_CS_FL == 1 & RUN1_DEST_ID == 1 | ST13 == 1 & RUN2_CS_
         "ST15_ST21" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 2),
         "ST15_ST18" = as.integer(ST13 == 1 & RUN2_SB_FL == 1 & RUN1_DEST_ID == 1),
         "ST16_ST19" = as.integer(ST16 == 1 & RUN1_SB_FL == 1 & RUN3_DEST_ID == 3),
         "ST16_ST11" = as.integer(ST16 == 1 & RUN1_CS_FL == 1 & RUN3_DEST_ID == 3),
         "ST17_ST20" = as.integer(ST17 == 1 & RUN1_SB_FL == 1 & RUN3_DEST_ID == 3),
         "ST17_ST12" = as.integer(ST17 == 1 & RUN1_CS_FL == 1 & RUN3_DEST_ID == 3),
         "ST18_END" = as.integer(ST18 == 1 & RUN1_CS_FL == 1),
         "ST18_ST22" = as.integer(ST18 == 1 & RUN1_SB_FL == 1 & RUN3_DEST_ID == 3)) %>%
  group_by(POS2_FLD_ID) %>%
  summarise("ST4_ST2" = sum(ST4_ST2),
            "ST4_ST7" = sum(ST4_ST7),
            "ST5_ST3" = sum(ST5_ST3),
            "ST5_ST8" = sum(ST5_ST8),
            "ST6_END" = sum(ST6_END),
            "ST6_ST9" = sum(ST6_ST9),
            "ST7_ST2" = sum(ST7_ST2),
            "ST7_ST10" = sum(ST7_ST10),
            "ST8_ST3" = sum(ST8_ST3),
            "ST8_ST11" = sum(ST8_ST11),
            "ST9_END" = sum(ST9_END),
            "ST9_ST12" = sum(ST9_ST12),
            "ST13_ST5" = sum(ST13_ST5),
            "ST13_ST8" = sum(ST13_ST8),
            "ST13_ST11" = sum(ST13_ST11),
            "ST13_ST16" = sum(ST13_ST16),
            "ST13_ST19" = sum(ST13_ST19),
            "ST14_ST6" = sum(ST14_ST6),
            "ST14_ST9" = sum(ST14_ST9),
            "ST14_ST12" = sum(ST14_ST12),
            "ST14_ST17" = sum(ST14_ST17),
            "ST14_ST20" = sum(ST14_ST20),
            "ST15_END" = sum(ST15_END),
            "ST15_ST18" = sum(ST15_ST18),
            "ST15_ST21" = sum(ST15_ST21),
            "ST16_ST11" = sum(ST16_ST11),
            "ST16_ST19" = sum(ST16_ST19),
            "ST17_ST12" = sum(ST17_ST12),
            "ST17_ST20" = sum(ST17_ST20),
            "ST18_END" = sum(ST18_END),
            "ST18_ST22" = sum(ST18_ST22))%>%
  mutate(Value = (RE[1,2]-RE[2,1])*ST4_ST2 +
           (RE[3,1]-RE[2,1])*ST4_ST7 +
           (RE[1,3]-RE[2,2])*ST5_ST3 +
           (RE[3,2]-RE[2,2])*ST5_ST8 +
           (0-RE[2,3])*ST6_END +
           (RE[3,3]-RE[2,3])*ST6_ST9 +
           (RE[3,1]-RE[1,2])*ST7_ST2 +
           (RE[4,1]-RE[3,1])*ST7_ST10 +
           (RE[1,3]-RE[3,2])*ST8_ST3 +
           (RE[4,2]-RE[3,2])*ST8_ST11+
           (0-RE[3,3])*ST9_END +
           (RE[4,3]-RE[3,3])*ST9_ST12 +
           (RE[2,2]-RE[5,1])*ST13_ST5 +
           (RE[3,2]-RE[5,1])*ST13_ST8 +
           (RE[4,2]-RE[5,1])*ST13_ST11 +
           (RE[6,1]-RE[5,1])*ST13_ST16 +
           (RE[7,1]-RE[5,1])*ST13_ST19 +
           (RE[2,3]-RE[5,2])*ST14_ST6 +
           (RE[3,3]-RE[5,2])*ST14_ST9 +
           (RE[4,3]-RE[5,2])*ST14_ST12 +
           (RE[6,2]-RE[5,2])*ST14_ST17 +
           (RE[7,2]-RE[5,2])*ST14_ST20 +
           (0-RE[5,3])*ST15_END +
           (RE[6,3]-RE[5,3])*ST15_ST18 +
           (RE[7,3]-RE[5,3])*ST15_ST21 +
           (RE[4,2]-RE[6,1])*ST16_ST11 +
           (RE[7,1]-RE[6,1])*ST16_ST19 +
           (RE[4,3]-RE[6,2])*ST17_ST12 +
           (RE[7,2]-RE[6,2])*ST17_ST20 +
           (0-RE[6,3])*ST18_END +
           (RE[7,3]-RE[6,3])*ST18_ST22,
         retroID = POS2_FLD_ID) %>%
  select(retroID, Value) %>%
  left_join(Names_Catchers) %>%
  select(Name, Value) %>%
  arrange(Value)
#Ten Best
Best_Catchers[1:10,]
```

```
## # A tibble: 10 x 2
##    Name               Value
##    <chr>              <dbl>
##  1 Jonathan Lucroy    -9.61
##  2 Salvador Perez     -9.43
##  3 James McCann       -7.87
##  4 Tucker Barnhart    -7.84
##  5 Jett Bandy         -7.28
##  6 J. T. Realmuto     -6.77
##  7 Buster Posey       -4.97
##  8 Matt Wieters       -4.65
##  9 Welington Castillo -4.6
## 10 Cameron Rupp       -4.34
```

```
#Ten Worst
Best_Catchers[104:95,]
```

```
## # A tibble: 10 x 2
##    Name           Value
##    <chr>          <dbl>
##  1 Tyler Flowers   5.44
##  2 Nick Hundley    3.76
##  3 Derek Norris    3.51
##  4 Kurt Suzuki     3.33
##  5 Yadier Molina   3.20
##  6 Russell Martin  2.76
##  7 Miguel Montero  2.56
##  8 Omar Narvaez    2.33
##  9 Hank Conger     2.32
## 10 Eric Fryer      2.06
```

For the purpose of this exercise, I opted to not look at any scenario where the runner on third base would be trying to take home. I don't think that that falls into the nature of the question about how effective a catcher is a preventing runs on attempted steals. Often steals of home are executed on the pitcher or are not related to a catcher's ability to throw runners out. The catchers with the lowest values are the best because that number represents the total change in run expectancy with that catcher.

- Compute break even stolen base percentages at different base out states and for different innings. You can consider a minimum attempt threshold. When are good times to attempt or not a stolen base?

```
dat2016 = dat2016 %>%
  mutate(RUNS = AWAY_SCORE_CT + HOME_SCORE_CT,
         HALF.INNING = paste(GAME_ID, INN_CT, BAT_HOME_ID),
         RUNS.SCORED = (BAT_DEST_ID > 3) + (RUN1_DEST_ID > 3) +
           (RUN2_DEST_ID > 3) + (RUN3_DEST_ID > 3))
half_innings = dat2016 %>%
  group_by(HALF.INNING) %>%
  summarise(Outs.Inning = sum(EVENT_OUTS_CT),
            Runs.Inning = sum(RUNS.SCORED),
            Runs.Start = first(RUNS),
            MAX.RUNS = Runs.Inning + Runs.Start)

dat2016 = dat2016 %>%
  inner_join(half_innings, by = "HALF.INNING") %>%
  mutate(RUNS.ROI = MAX.RUNS - RUNS)

dat2016 = dat2016 %>%
  mutate(BASES = paste(ifelse(BASE1_RUN_ID != "",1,0),
                       ifelse(BASE2_RUN_ID != "",1,0),
                       ifelse(BASE3_RUN_ID != "",1,0), sep = ""),
         STATE = paste(BASES, OUTS_CT))

dat2016 = dat2016 %>%
  mutate(NRUNNER1 = as.numeric(RUN1_DEST_ID == 1 | BAT_DEST_ID == 1),
         NRUNNER2 = as.numeric(RUN1_DEST_ID == 2 | RUN2_DEST_ID == 2 | BAT_DEST_ID == 2),
         NRUNNER3 = as.numeric(RUN1_DEST_ID == 3 | RUN2_DEST_ID == 3 |
                                 RUN3_DEST_ID == 3 | BAT_DEST_ID == 3),
         NOUTS = OUTS_CT + EVENT_OUTS_CT,
         NEW.BASES = paste(NRUNNER1, NRUNNER2, NRUNNER3, sep = ""),
         NEW.STATE = paste(NEW.BASES, NOUTS)) %>%
  filter((STATE != NEW.STATE) | (RUNS.SCORED > 0)) %>%
  filter(Outs.Inning == 3)

RUNS = dat2016 %>%
  group_by(STATE) %>%
  summarize(Mean = mean(RUNS.ROI)) %>%
  mutate(Outs = substr(STATE, 5, 5)) %>%
  arrange(Outs)

RUNS_out = matrix(round(RUNS$Mean, 2), 8, 3)
dimnames(RUNS_out)[[1]] = c("000","001","010","011",
                            "100","101","110","111")
dimnames(RUNS_out)[[2]] = c("0 outs", "1 out", "2 outs")

dat2016 = dat2016 %>%
  left_join(select(RUNS, - Outs), by = "STATE") %>%
  rename(Runs.State = Mean) %>%
  left_join(select(RUNS, -Outs), by = c("NEW.STATE" = "STATE")) %>%
  rename(Runs.New.State = Mean) %>%
  replace_na(list(Runs.New.State = 0)) %>%
  mutate(run_value = Runs.New.State - Runs.State + RUNS.SCORED)

stealing = dat2016 %>% filter(EVENT_CD %in% c(4,6))
steal_states = stealing %>%
  group_by(INN_CT, STATE) %>%
  summarise(N = n()) %>%
  filter(N >= 10) %>%
  select(STATE)
steal_matrix = stealing %>%
  group_by(EVENT_CD, INN_CT, STATE) %>%
  summarise(N = n(),
            avg_run_value = mean(run_value)) %>%
  mutate(pct = N/sum(N)) %>%
  arrange(INN_CT, STATE)
steal_final = steal_states %>%
  left_join(steal_matrix)

#Removing observations without data for stealing and caught stealing
steal_final = steal_final[-c(11,84,91),]
inn_state = steal_final %>%
  ungroup(EVENT_CD)%>%
  filter(N > 0) %>%
  select(STATE, INN_CT) %>%
  unique()
#Computing break-even percentage from slides
break_even = steal_final[seq(2, 110, 2), 5]/(steal_final[seq(1,109,2),5]-steal_final[seq(2,110,2),5])*-
```

```
final = cbind.data.frame(inn_state, break_even) %>%
  mutate(break_even = avg_run_value) %>%
  select(-avg_run_value) %>%
  arrange(break_even)

head(final)
```

```
##   STATE INN_CT break_even
## 1 101 2      2 -0.1944789
## 2 101 1      2  0.3559485
## 3 110 1      1  0.5935125
## 4 110 1      8  0.6025332
## 5 110 1      4  0.6296851
## 6 010 1      4  0.6317965
```

According to this, the best times to steal a base with the formula provided in the notes would be in the second inning with runners on the corners and two outs. In fact, according to this, it would be harmful if you did not try to steal. This list shows the lowest break-even percentages. To interpret this list in the context of the question, these would be the best times to steal as they have the least risk.

Question 2 In the Simulation Notes we considered team specific transition probabilities for one base out state corresponding to the St. Louis Cardinals in 2016. Do the following:

- Build the entire transition probability matrix for the St. Louis Cardinals. This matrix should be constructed using the normalized versions of the reliable probabilities with $K = 1274$. Hint: you may want to use the code in the notes within a loop, and then convert from long format to wide format using something like

```
library("reshape2")
dcast(melt(foo, id.vars=c("STATE", "NEW.STATE")), STATE~NEW.STATE) %>%
    replace(is.na(.), 0)
```

```
dat2016 = read_csv("C:/Users/STP/Desktop/stat430/stat430sp25/all2016.csv",
                   col_names = pull(fields, Header),
                   na = character())
dat2016 = dat2016 %>%
  mutate(RUNS  = AWAY_SCORE_CT + HOME_SCORE_CT,
         HALF.INNING = paste(GAME_ID, INN_CT, BAT_HOME_ID),
         RUNS.SCORED = (BAT_DEST_ID > 3) + (RUN1_DEST_ID > 3) +
           (RUN2_DEST_ID > 3) + (RUN3_DEST_ID > 3))

half_innings = dat2016 %>%
  group_by(HALF.INNING) %>%
  summarize(Outs.Inning = sum(EVENT_OUTS_CT),
            Runs.Inning = sum(RUNS.SCORED),
            Runs.Start = first(RUNS),
            MAX.RUNS = Runs.Inning + Runs.Start)
dat2016 = dat2016 %>%
  inner_join(half_innings, by = "HALF.INNING") %>%
  mutate(BASES = paste(ifelse(BASE1_RUN_ID > '', 1, 0),
                       ifelse(BASE2_RUN_ID > '', 1, 0),
                       ifelse(BASE3_RUN_ID > '', 1, 0), sep = ""),
         STATE = paste(BASES, OUTS_CT),
         NRUNNER1 = as.numeric(RUN1_DEST_ID == 1 | BAT_DEST_ID == 1),
         NRUNNER2 = as.numeric(RUN1_DEST_ID == 2 | RUN2_DEST_ID == 2 | BAT_DEST_ID == 2),
         NRUNNER3 = as.numeric(RUN1_DEST_ID == 3 | RUN2_DEST_ID == 3 |
                               RUN3_DEST_ID == 3 | BAT_DEST_ID == 3),
         NOUTS = OUTS_CT + EVENT_OUTS_CT,
         NEW.BASES = paste(NRUNNER1, NRUNNER2, NRUNNER3, sep = ""),
         NEW.STATE = paste(NEW.BASES, NOUTS))

dat2016 = dat2016 %>% filter((STATE != NEW.STATE) | (RUNS.SCORED > 0))
dat2016C = dat2016 %>% filter(Outs.Inning == 3, BAT_EVENT_FL == TRUE)
dat2016C = dat2016C %>% mutate(NEW.STATE = gsub("[0-1]{3} 3", "3", NEW.STATE))

T_matrix = dat2016C %>%
  select(STATE, NEW.STATE) %>%
  table()

P_matrix = prop.table(T_matrix, 1)

P_matrix = rbind(P_matrix, c(rep(0, 24), 1))


P_matrix_3 = P_matrix %*% P_matrix %*% P_matrix

#League averages
P_matrix_3 %>% as_tibble(rownames = "STATE") %>%
  filter(STATE == "000 0") %>%
  gather(key = "NEW.STATE", value = "Prob", -STATE) %>%
  arrange(desc(Prob)) %>%
  head()
```

```
## # A tibble: 6 x 3
##   STATE NEW.STATE   Prob
##   <chr> <chr>      <dbl>
## 1 000 0 3         0.372
## 2 000 0 100 2     0.241
## 3 000 0 110 1     0.0815
## 4 000 0 010 2     0.0739
## 5 000 0 000 2     0.0529
## 6 000 0 001 2     0.0286
```

```
dat2016C = dat2016C %>%
  mutate(HOME_TEAM_ID = str_sub(GAME_ID, 1, 3),
         BATTING.TEAM = ifelse(BAT_HOME_ID == 0,
                               AWAY_TEAM_ID, HOME_TEAM_ID))

Team.T = dat2016C %>%
  group_by(BATTING.TEAM, STATE, NEW.STATE) %>%
  count()

Team.T.S = dat2016C %>%
  group_by(BATTING.TEAM, STATE, NEW.STATE) %>% tally()
```

```r
SLN.Trans = Team.T.S %>% filter(BATTING.TEAM == "SLN") %>%
  mutate(p = n / sum(n))

All.Trans = dat2016C %>%
  group_by(NEW.STATE) %>% tally() %>%
  mutate(p = n / sum(n))

SLN_FULL_TRANS = SLN.Trans %>% inner_join(All.Trans, by = "NEW.STATE") %>%
  mutate(p.EST = n.x / (1274 + n.x) * p.x + 1274 / (1274 + n.x) * p.y) %>%
  mutate(p.EST = p.EST / sum(p.EST)) %>%
  select(BATTING.TEAM, NEW.STATE, p.x, p.y, p.EST)
SLN_FULL_TRANS
```

```
## # A tibble: 191 x 6
## # Groups:   BATTING.TEAM, STATE [24]
##    STATE BATTING.TEAM NEW.STATE     p.x     p.y   p.EST
##    <chr> <chr>        <chr>       <dbl>   <dbl>   <dbl>
##  1 000 0 SLN          000 0     0.0419  0.0114  0.0247
##  2 000 0 SLN          000 1     0.666   0.177   0.757
##  3 000 0 SLN          001 0     0.00786 0.00190 0.00376
##  4 000 0 SLN          010 0     0.0629  0.0147  0.0347
##  5 000 0 SLN          100 0     0.221   0.0594  0.180
##  6 000 1 SLN          000 1     0.0338  0.177   0.270
##  7 000 1 SLN          000 2     0.668   0.141   0.521
##  8 000 1 SLN          001 1     0.00730 0.00826 0.0129
##  9 000 1 SLN          010 1     0.0511  0.0245  0.0399
## 10 000 1 SLN          100 1     0.240   0.0713  0.156
## # i 181 more rows
```

- Display the typical states after 3 PAs to start a half-inning for the 2016 St. Louis Cardinals. Were the 2016 Cardinals expected to end a half-inning more frequently than the average team after 3 PAs?

```r
SLNC = dat2016C %>%
  filter(AWAY_TEAM_ID == "SLN" & BAT_HOME_ID == 0 | str_detect(GAME_ID, "^SLN")  & BAT_HOME_ID == 1)

T_SLN = SLNC %>%
  select(STATE, NEW.STATE) %>%
  table()
P_SLN = prop.table(T_SLN, 1)

P_SLN = rbind(P_SLN, c(rep(0, 24), 1))

Final_SLN = P_SLN %*% P_SLN %*% P_SLN

Final_SLN %>%
  as_tibble(rownames = "STATE") %>%
  filter(STATE == "000 0") %>%
  gather(key = "NEW.STATE", value = "Prob", -STATE) %>%
  arrange(desc(Prob))
```

```
## # A tibble: 25 x 3
##    STATE NEW.STATE   Prob
##    <chr> <chr>      <dbl>
##   1 000 0 3         0.352
##   2 000 0 100 2     0.262
##   3 000 0 110 1     0.0764
##   4 000 0 010 2     0.0673
##   5 000 0 000 2     0.0635
##   6 000 0 001 2     0.0333
##   7 000 0 100 1     0.0300
##   8 000 0 101 1     0.0250
##   9 000 0 000 1     0.0178
##  10 000 0 010 1     0.0174
## # i 15 more rows
```

The average for the league in 2016 was that the probability of ending an inning after three PAs was 0.372. The Cardinals' probability for ending the inning after three PAs was 0.352. They ended their innings less frequently than the league did on average.

- Use the `simulate_half_inning` function and the Runs matrix `R` in the notes to construct a 2016 Cardinals specific RE24 matrix.

```r
count_runners_out = function(s){
  s %>% str_split("") %>% pluck(1) %>% as.numeric() %>% sum(na.rm = TRUE)
}

num_havent_scored <- function(s) {
  s |>
    str_split("") |>
    pluck(1) |>
    as.numeric() |>
    sum(na.rm = TRUE)
}

runners_out <- T_SLN |>
  row.names() |>
  set_names() |>
  map_int(num_havent_scored)


R_SLN <- outer(
  runners_out + 1,
  runners_out,
  FUN = "-"
) |>
  cbind("3" = rep(0, 24))


simulate_half_inning <- function(P, R, start = 1) {
  s <- start
  path <- NULL
  runs <- 0
  while (s < 25) {
    s_new <- sample(1:25, size = 1, prob = P[s, ])
    path <- c(path, s_new)
    runs <- runs + R[s, s_new]
    s <- s_new
  }
  runs
}

runs_j <- function(j) {
  1:10000 |>
    map_int(~simulate_half_inning(T_SLN, R_SLN, j)) |>
    mean()
}

erm_SLN_mc <- tibble(
  state = row.names(T_SLN),
  mean_run_value = map_dbl(1:24, runs_j)
) |>
  mutate(
    bases = str_sub(state, 1, 3),
    outs_ct = as.numeric(str_sub(state, 5, 5))
  ) |>
  select(-state)

erm_SLN_mc |>
  pivot_wider(names_from = outs_ct, values_from = mean_run_value)
```

```
## # A tibble: 8 x 4
##   bases  `0`   `1`   `2`
##   <chr> <dbl> <dbl> <dbl>
## 1 000   0.511 0.278 0.116
## 2 001   1.34  1.03  0.413
## 3 010   1.20  0.757 0.318
## 4 011   1.95  1.60  0.548
## 5 100   0.835 0.498 0.194
## 6 101   1.92  1.43  0.527
## 7 110   1.74  1.00  0.382
## 8 111   2.86  1.78  0.613
```

I did it closer to the way from the book rather than the way from the slides due to the "doMC" package not being able to install on my computer. This is still the expected runs for each of the 24 states for the Cardinals in 2016.

Question 3 Problem 5 in Section 5.11 of Analyzing Baseball Data with R.

```
#a
singles = dat2016 %>%
  filter(EVENT_CD == 20)

#b
part_b_table = singles %>%
  group_by(STATE, NEW.STATE) %>%
  summarise(Freq = n(), RUN = sum(RUNS.SCORED != 0)) %>%
  mutate(RUN = RUN/RUN)
part_b_table = part_b_table %>%
  replace(is.na(part_b_table), 0)
part_b_table
```

```
## # A tibble: 192 x 4
## # Groups:   STATE [24]
##    STATE NEW.STATE  Freq   RUN
##    <chr> <chr>     <int> <dbl>
##  1 000 0 000 1        55     0
##  2 000 0 001 0         8     0
##  3 000 0 010 0        81     0
##  4 000 0 100 0      6776     0
##  5 000 1 000 2        44     0
##  6 000 1 001 1         5     0
##  7 000 1 010 1        59     0
##  8 000 1 100 1      4659     0
##  9 000 2 000 3        40     0
## 10 000 2 001 2         2     0
## # i 182 more rows
```

```
#c
part_b_table %>%
  filter(str_detect(STATE, "100")) %>%
  filter(str_detect(NEW.STATE, "101") | str_detect(NEW.STATE, "110") | str_detect(NEW.STATE, "011"))
```

```
## # A tibble: 9 x 4
## # Groups:   STATE [3]
##   STATE NEW.STATE  Freq   RUN
##   <chr> <chr>     <int> <dbl>
## 1 100 0 011 0        30     0
## 2 100 0 101 0       406     0
## 3 100 0 110 0      1194     0
## 4 100 1 011 1        29     0
## 5 100 1 101 1       468     0
## 6 100 1 110 1      1418     0
## 7 100 2 011 2        21     0
## 8 100 2 101 2       491     0
## 9 100 2 110 2      1217     0
```

```
#d
part_b_table %>%
  filter(str_detect(STATE, "110")) %>%
  group_by(STATE) %>%
  summarise(R_Prob = round(sum(RUN) / sum(Freq), 3))
```

```
## # A tibble: 3 x 2
##   STATE R_Prob
##   <chr>  <dbl>
## 1 110 0  0.022
## 2 110 1  0.013
## 3 110 2  0.011
```

For each of the possible starting states, here are the probabilities a run is scored on the play. I simply took the total number of plays a run scored and divided it by the total number of plays starting with that baseout state.

Question 4 Problem 1 in Section 9.5 of Analyzing Baseball Data with R.

```
P <- matrix(c(.3, .7,  0,  0,
               0, .3, .7,  0,
               0,  0, .3, .7,
               0,  0,  0,  1), 4, 4, byrow = TRUE)

P2 <- P %*% P

P2
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 0.09 0.42 0.49 0.00
## [2,] 0.00 0.09 0.42 0.49
## [3,] 0.00 0.00 0.09 0.91
## [4,] 0.00 0.00 0.00 1.00
```

```
N <- solve(diag(c(1, 1, 1)) - P[-4, -4])

sum(N[1,1:3])
```

```
## [1] 4.285714
```

a. The probability of moving from 0 outs to 1 out after 2 PAs in this model is 0.42.
b. The average number of plate appearances in this model is 4.286. Interestingly, this matrix has the same number of average appearances at each out state.

Question 5 Problem 3 in Section 9.5 of Analyzing Baseball Data with R.

```
library(tidyverse)
dat1968 = read_csv("C:/Users/STP/Desktop/stat430/stat430sp25/all2016.csv",
                     col_names = pull(fields, Header),
                    na = character())
dat1968 = dat1968 %>%
  mutate(RUNS  = AWAY_SCORE_CT + HOME_SCORE_CT,
         HALF.INNING = paste(GAME_ID, INN_CT, BAT_HOME_ID),
         RUNS.SCORED = (BAT_DEST_ID > 3) + (RUN1_DEST_ID > 3) +
           (RUN2_DEST_ID > 3) + (RUN3_DEST_ID > 3))

half_innings2 = dat1968 %>%
  group_by(HALF.INNING) %>%
  summarize(Outs.Inning = sum(EVENT_OUTS_CT),
            Runs.Inning = sum(RUNS.SCORED),
            Runs.Start = first(RUNS),
            MAX.RUNS = Runs.Inning + Runs.Start)
dat1968 = dat1968 %>%
  inner_join(half_innings2, by = "HALF.INNING") %>%
  mutate(BASES = paste(ifelse(BASE1_RUN_ID > '', 1, 0),
                       ifelse(BASE2_RUN_ID > '', 1, 0),
                       ifelse(BASE3_RUN_ID > '', 1, 0), sep = ""),
         STATE = paste(BASES, OUTS_CT),
         NRUNNER1 = as.numeric(RUN1_DEST_ID == 1 | BAT_DEST_ID == 1),
         NRUNNER2 = as.numeric(RUN1_DEST_ID == 2 | RUN2_DEST_ID == 2 | BAT_DEST_ID == 2),
         NRUNNER3 = as.numeric(RUN1_DEST_ID == 3 | RUN2_DEST_ID == 3 |
                               RUN3_DEST_ID == 3 | BAT_DEST_ID == 3),
         NOUTS = OUTS_CT + EVENT_OUTS_CT,
         NEW.BASES = paste(NRUNNER1, NRUNNER2, NRUNNER3, sep = ""),
         NEW.STATE = paste(NEW.BASES, NOUTS))

dat1968 = dat1968 %>% filter((STATE != NEW.STATE) | (RUNS.SCORED > 0))
datC = dat1968 %>% filter(Outs.Inning == 3, BAT_EVENT_FL == TRUE)
datC = datC %>% mutate(NEW.STATE = gsub("[0-1]{3} 3", "3", NEW.STATE))

T_matrix2 = datC %>%
  select(STATE, NEW.STATE) %>%
  table()

P_matrix2 = prop.table(T_matrix2, 1)

P_matrix2 = rbind(P_matrix2, c(rep(0, 24), 1))


count_runners_out = function(s){
  s %>% str_split("") %>% pluck(1) %>% as.numeric() %>% sum(na.rm = TRUE)
}

num_havent_scored <- function(s) {
  s |>
    str_split("") |>
    pluck(1) |>
    as.numeric() |>
    sum(na.rm = TRUE)
}

runners_out <- T_matrix2 |>
  row.names() |>
  set_names() |>
  map_int(num_havent_scored)


R_runs <- outer(
  runners_out + 1,
  runners_out,
  FUN = "-"
) |>
  cbind("3" = rep(0, 24))


simulate_half_inning <- function(P, R, start = 1) {
  s <- start
  path <- NULL
  runs <- 0
  while (s < 25) {
    s_new <- sample(1:25, size = 1, prob = P[s, ])
    path <- c(path, s_new)
    runs <- runs + R[s, s_new]
    s <- s_new
  }
  runs
}

runs_j <- function(j) {
  1:10000 |>
    map_int(~simulate_half_inning(T_matrix2, R_runs, j)) |>
    mean()
}

erm_1968_mc <- tibble(
  state = row.names(T_matrix2),
  mean_run_value = map_dbl(1:24, runs_j)
) |>
  mutate(
    bases = str_sub(state, 1, 3),
    outs_ct = as.numeric(str_sub(state, 5, 5))
  ) |>
  select(-state)

erm_1968_mc |>
  pivot_wider(names_from = outs_ct, values_from = mean_run_value)
```

```
## # A tibble: 8 x 4
##   bases   `0`   `1`   `2`
##   <chr> <dbl> <dbl> <dbl>
## 1 000   0.469 0.264 0.103
## 2 001   1.29  0.917 0.346
## 3 010   1.12  0.638 0.295
## 4 011   1.89  1.31  0.496
## 5 100   0.809 0.505 0.209
## 6 101   1.70  1.14  0.442
## 7 110   1.41  0.873 0.384
## 8 111   2.17  1.46  0.690
```

These numbers are overall very similar to the 2016 numbers. That can make sense to me because scoring had gone up from this period until about the 2000s, and then pitching started to become very, very good. The expected value from the start of the inning is a little lower here than in 2016, but it looks like the further into the inning teams get, the more likely they are to score in 1968 than in 2016.