BHao_Assign3

Problem 3.5.1

a) As the rolls increase, the observed results approach the calculated actual probabilities.

```r
set.seed(123)

# create simulation function
roll_dice = function(rolls) {
  tbl_results = data.table(roll = seq(1:rolls),
                           dice1 = sample(seq(1:6), rolls, replace = TRUE),
                           dice2 = sample(seq(1:6), rolls, replace = TRUE))
  tbl_results[, dice1_2 := dice1 + dice2]
}

roll_dice_50 = roll_dice(50)
roll_dice_500 = roll_dice(500)
roll_dice_10000 = roll_dice(10000)

data.table(rolls = c('Actual probality', 50, 500, 10000),
           rbind(c(1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1)/36,
                 prop.table(table(roll_dice_50$dice1_2)),
                 prop.table(table(roll_dice_500$dice1_2)),
                 prop.table(table(roll_dice_10000$dice1_2))))
```

```
## Warning in rbind(c(1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1)/36,
## prop.table(table(roll_dice_50$dice1_2)), : number of columns of result is
## not a multiple of vector length (arg 2)
```

```
##               rolls          2          3          4         5         6
## 1: Actual probality 0.02777778 0.05555556 0.08333333 0.1111111 0.1388889
## 2:               50 0.06000000 0.08000000 0.06000000 0.0600000 0.1200000
## 3:              500 0.03000000 0.05600000 0.09200000 0.1020000 0.1620000
## 4:            10000 0.02810000 0.05550000 0.08240000 0.1157000 0.1482000
##            7         8         9         10         11         12
## 1: 0.1666667 0.1388889 0.1111111 0.08333333 0.05555556 0.02777778
## 2: 0.2000000 0.1600000 0.0800000 0.10000000 0.08000000 0.06000000
## 3: 0.1600000 0.1400000 0.0840000 0.08800000 0.06200000 0.02400000
## 4: 0.1614000 0.1380000 0.1116000 0.07830000 0.05390000 0.02690000
```

b) Here, we've loaded both die to make 1s three times more likely to appear than other numbers.

```r
set.seed(123)

# create simulation function
roll_loaded_dice = function(rolls) {
  tbl_results = data.table(roll = seq(1:rolls),
                           # load dice by making 1s three times more likely to appear than other number
                           dice1 = sample(c(1, 1, seq(1:6)), rolls, replace = TRUE),
                           dice2 = sample(c(1, 1, seq(1:6)), rolls, replace = TRUE))
  tbl_results[, dice1_2 := dice1 + dice2]
}
```

```
roll_loaded_dice_50 = roll_loaded_dice(50)
roll_loaded_dice_500 = roll_loaded_dice(500)
roll_loaded_dice_10000 = roll_loaded_dice(10000)

# calculate actual probabilites
actual_probs = data.frame(dice = c(1, 1, seq(1:6)))
actual_probs = actual_probs %>% merge(actual_probs, by = NULL) %>%
  mutate(dice1_2 = dice.x + dice.y) %>%
  select(dice1_2)

data.table(rolls = c('Actual probality', 50, 500, 10000),
           rbind(prop.table(table(actual_probs)),
                 prop.table(table(roll_loaded_dice_50$dice1_2)),
                 prop.table(table(roll_loaded_dice_500$dice1_2)),
                 prop.table(table(roll_loaded_dice_10000$dice1_2))))
```

```
## Warning in rbind(prop.table(table(actual_probs)),
## prop.table(table(roll_loaded_dice_50$dice1_2)), : number of columns of
## result is not a multiple of vector length (arg 2)
```

```
##                rolls        2       3        4      5        6       7
## 1: Actual probality 0.140625 0.09375 0.109375 0.1250 0.140625 0.15625
## 2:                50 0.180000 0.10000 0.060000 0.0600 0.200000 0.14000
## 3:               500 0.152000 0.09400 0.102000 0.1400 0.160000 0.12000
## 4:             10000 0.137900 0.10240 0.107900 0.1251 0.150000 0.14970
##           8      9       10      11       12
## 1: 0.078125 0.0625 0.046875 0.03125 0.015625
## 2: 0.100000 0.0400 0.060000 0.06000 0.180000
## 3: 0.072000 0.0620 0.052000 0.03200 0.014000
## 4: 0.078000 0.0587 0.045400 0.03030 0.014600
```

c) See answer to a) above.

Problem 3.5.5

Using 90, 95 and 99% confidence intervals and 1000 runs, I generally saw results within +/- 1-3% from the stated confidence interval.

```
# create function for simulation
integrate_sim = function(mu, sig, a, b, n, ci) {
  tbl_results = data.table(sim = seq(1:n), Xi = runif(n) * (b - a ) + a)
  tbl_results[, est := (b - a) * dnorm(Xi, mu, sig)]
  tbl_results = tbl_results[, .(mean = mean(est), se = sd(est) / sqrt(n))]
  ci = qnorm(mean(c(ci, 1)))
  return(list(lowerCI = tbl_results$mean - ci * tbl_results$se,
              upperCI = tbl_results$mean + ci * tbl_results$se))
}

exact = (pnorm(6.7, 5.8, 2.3) - pnorm(4.5, 5.8, 2.3))
df = data.frame(ci_90 = rep(0, 100), ci_95 = rep(0, 100), ci_99 = rep(0, 100))
for (i in 1:1000) {
  c = 1
  for (test in c(0.90, 0.95, 0.99)) {
    ci = integrate_sim(5.8, 2.3, 4.5, 6.7, 50, test)
```

```
    df[i, c] = ci$lowerCI < exact & exact < ci$upperCI
    c = c + 1
  }
}
colSums(df)/1000
```

```
## ci_90 ci_95 ci_99
## 0.881 0.932 0.972
```

Problem 3.5.17

```
set.seed(123)

# create simulation function for one season
run_sim = function(days = 90) {
  # create data table to house inputs
  tbl_items = data.table(item = c('oats', 'peas', 'beans', 'barley'),
                         cost = c(1.05, 3.17, 1.99, 0.95),
                         price = c(1.29, 3.76, 2.23, 1.65),
                         minq = c(0, 0, 0, 0),
                         maxq = c(10, 8, 14, 11))
  tbl_items = tbl_items[, profit := price - cost]

  # create data frame to house results
  tbl_sim = data.frame(day = rep(0, days), total_revenue = rep(0, days),
                       total_cost = rep(0, days), total_profit = rep(0, days))

  for (i in 1:days) {
    qty = NULL
    revenue = 0
    cost = 0
    profit = 0
    for (r in 1:nrow(tbl_items)) {
      qty = sample(tbl_items[r, minq]:tbl_items[r, maxq], 1)
      revenue = revenue + tbl_items[r, price] * qty
      cost = cost + tbl_items[r, cost] * qty
      profit = profit + tbl_items[r, profit] * qty
    }
    tbl_sim[i,] = c(i, revenue, cost, profit)
  }
  return(data.table(tbl_sim))
}

# create simulation function for x seasons
run_sim_season = function(seasons = 1000, days = 90) {
  tbl_daily = data.frame(day = rep(0, seasons*days), total_revenue = rep(0, seasons*days),
                         total_cost = rep(0, seasons*days), total_profit = rep(0, seasons*days))
  tbl_season = data.frame(season = rep(0, seasons), total_revenue = rep(0, seasons),
                          total_cost = rep(0, seasons), total_profit = rep(0, seasons))
  for (i in 1:seasons) {
    ## REPLACE LOGIC BELOW WITH A SIMPLE ARRAY AND THEN RETURN A DATA.TABLE
    sim = run_sim(days = days)
```

```r
    tbl_daily[(i*days-days+1):(i*days),] = sim
    tbl_season[i,] = c(season = i, sim[, lapply(.SD, sum), .SDcols = 2:4])
  }
  return(list(tbl_daily = data.table(tbl_daily), tbl_season = data.table(tbl_season)))
}

# run simulation x times and aggregate results
sim_results = run_sim_season(seasons = 1, days = 90)

summary(sim_results$tbl_daily)
```

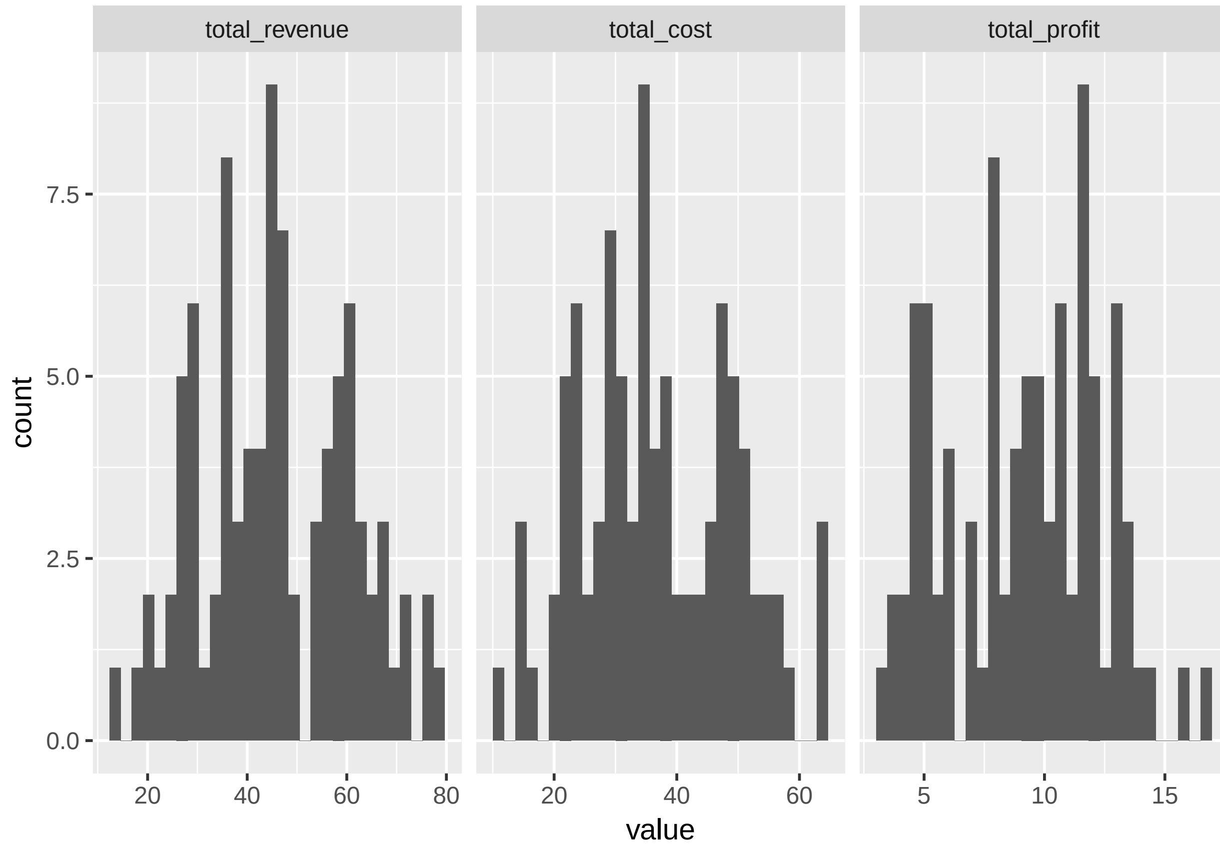
```
##       day        total_revenue     total_cost     total_profit
##  Min.   : 1.00   Min.   :14.58   Min.   :11.41   Min.   : 3.170
##  1st Qu.:23.25   1st Qu.:35.08   1st Qu.:28.20   1st Qu.: 6.402
##  Median :45.50   Median :45.05   Median :34.98   Median : 9.365
##  Mean   :45.50   Mean   :45.86   Mean   :36.69   Mean   : 9.167
##  3rd Qu.:67.75   3rd Qu.:58.91   3rd Qu.:46.91   3rd Qu.:11.640
##  Max.   :90.00   Max.   :79.67   Max.   :64.25   Max.   :16.630
```

```r
summary(sim_results$tbl_season)
```

```
##      season  total_revenue    total_cost    total_profit
##  Min.   :1   Min.   :4127   Min.   :3302   Min.   :825.1
##  1st Qu.:1   1st Qu.:4127   1st Qu.:3302   1st Qu.:825.1
##  Median :1   Median :4127   Median :3302   Median :825.1
##  Mean   :1   Mean   :4127   Mean   :3302   Mean   :825.1
##  3rd Qu.:1   3rd Qu.:4127   3rd Qu.:3302   3rd Qu.:825.1
##  Max.   :1   Max.   :4127   Max.   :3302   Max.   :825.1
```

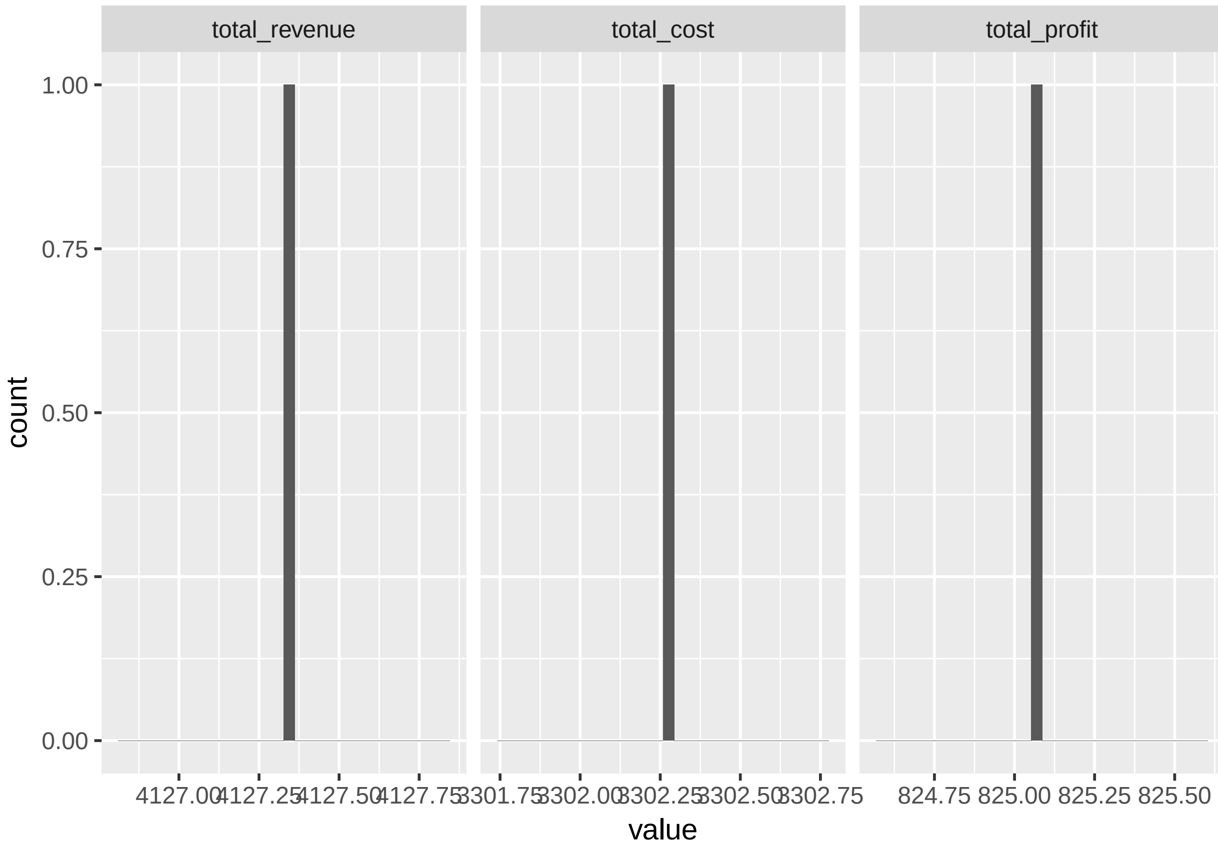
```r
sim_results$tbl_daily %>% select(-day) %>%
  gather('metric', 'value') %>%
  mutate('metric' = factor(metric, levels = c('total_revenue', 'total_cost', 'total_profit'))) %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_grid(. ~ metric, scales = 'free')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
sim_results$tbl_season %>% select(-season) %>%
  gather('metric', 'value') %>%
  mutate('metric' = factor(metric, levels = c('total_revenue', 'total_cost', 'total_profit'))) %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_grid(. ~ metric, scales = 'free')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


total_revenue
total_cost
total_profit
count
1.00
0.75
0.50
0.25
0.00
4127.00
4127.25
4127.50
4127.75
3301.75
3302.00
3302.25
3302.50
3302.75
824.75
825.00
825.25
825.50
value