DATA 604 Assignment 3: Types of Simulation

Dan Smilowitz

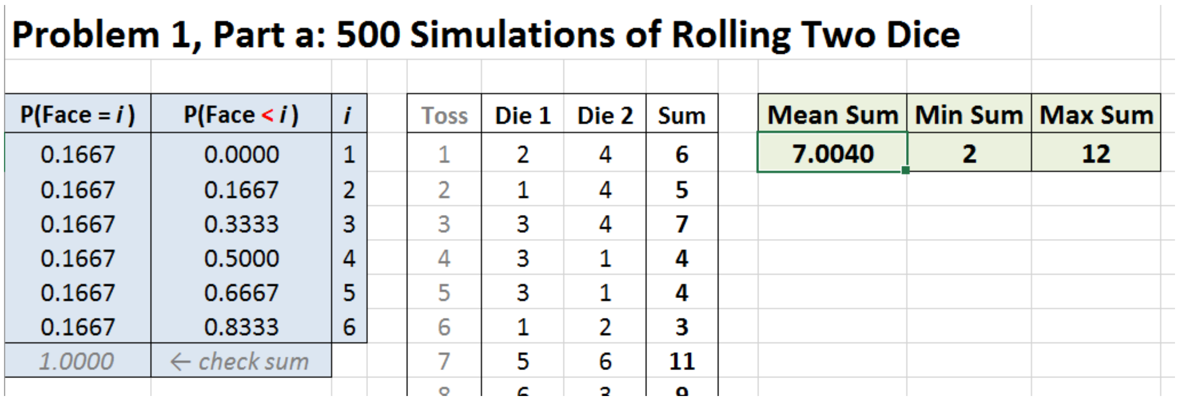
February 14, 2017

Problem 1

The extensions to Model 3-1 are performed in the included 'Smilowitz_wk3.xlsx' file.

Part a

Included in the tab '1a', the number of simulations is expanded to 500 rows.



Problem 1, Part a: 500 Simulations of Rolling Two Dice

P(Face = *i*)	P(Face < *i*)	*i*
0.1667	0.0000	1
0.1667	0.1667	2
0.1667	0.3333	3
0.1667	0.5000	4
0.1667	0.6667	5
0.1667	0.8333	6
1.0000	*← check sum*	

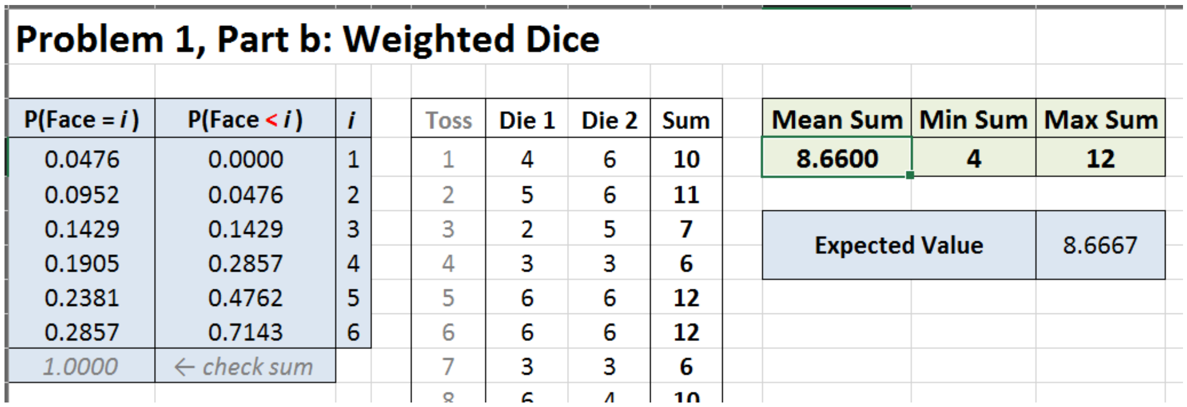
Toss	Die 1	Die 2	Sum
1	2	4	6
2	1	4	5
3	3	4	7
4	3	1	4
5	3	1	4
6	1	2	3
7	5	6	11

Mean Sum	Min Sum	Max Sum
7.0040	2	12

Figure 1:

Part b

Reverting to 50 simulations, the dice are assigned a new weighting where the probability of a value is proportional to the value. The new expected value of this weighted pair of dice is 8.6667. The calculation of weights and results are in tab '1b'.



Problem 1, Part b: Weighted Dice

P(Face = *i*)	P(Face < *i*)	*i*
0.0476	0.0000	1
0.0952	0.0476	2
0.1429	0.1429	3
0.1905	0.2857	4
0.2381	0.4762	5
0.2857	0.7143	6
1.0000	*← check sum*	

Toss	Die 1	Die 2	Sum
1	4	6	10
2	5	6	11
3	2	5	7
4	3	3	6
5	6	6	12
6	6	6	12
7	3	3	6

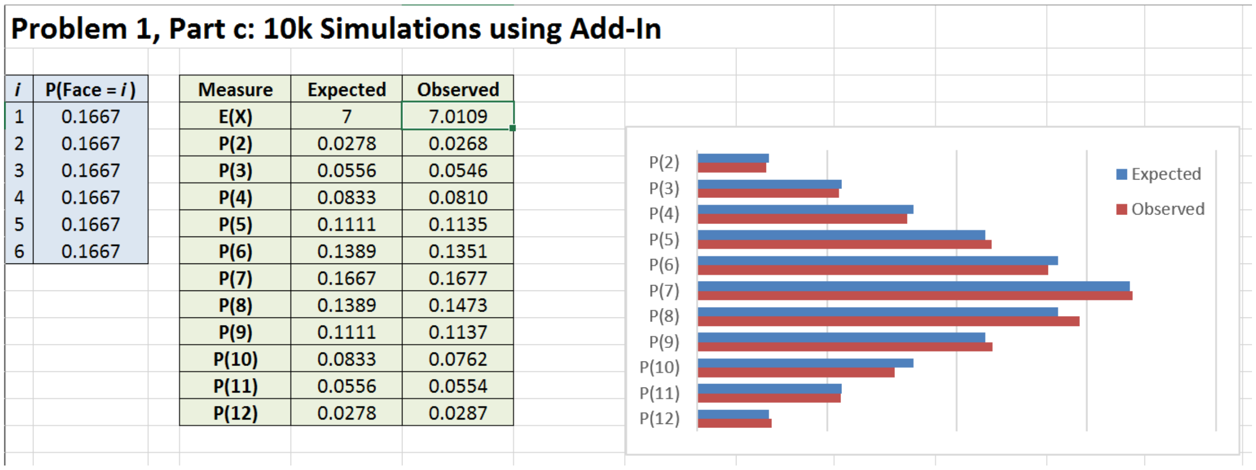
Mean Sum	Min Sum	Max Sum
8.6600	4	12

Expected Value	8.6667

Figure 2:

Part c

Two sets of 10,000 samples for fair dice (one for each die) are created using the "Random Number Generator" function of Excel's *Data Analysis* add-in – the generated data can be viewed in the 'Data' tab of the workbook. The observed average and distribution of sums are calculated and compared to the expected values in the '1c' tab.



Problem 1, Part c: 10k Simulations using Add-In

i	P(Face = *i*)
1	0.1667
2	0.1667
3	0.1667
4	0.1667
5	0.1667
6	0.1667

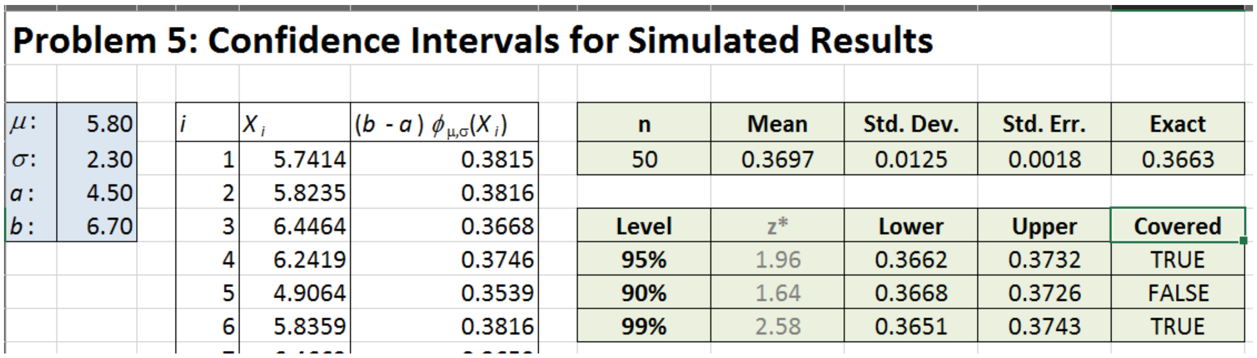
Measure	Expected	Observed
E(X)	7	7.0109
P(2)	0.0278	0.0268
P(3)	0.0556	0.0546
P(4)	0.0833	0.0810
P(5)	0.1111	0.1135
P(6)	0.1389	0.1351
P(7)	0.1667	0.1677
P(8)	0.1389	0.1473
P(9)	0.1111	0.1137
P(10)	0.0833	0.0762
P(11)	0.0556	0.0554
P(12)	0.0278	0.0287

Figure 3:

Problem 5

As shown in tab '5', the standard deviation of the simulated results are calculated. This value is used to calculate the standard error $se = s/\sqrt{n}$ for the sample. For each of the required confidence levels, the associated z-score z^* is calculated. These values are then used to calculate the confidence intervals:

$$CI = \bar{x} \pm se \times z^*$$



Problem 5: Confidence Intervals for Simulated Results

μ:	5.80
σ:	2.30
a:	4.50
b:	6.70

i	X_i	$(b-a)\,\phi_{\mu,\sigma}(X_i)$
1	5.7414	0.3815
2	5.8235	0.3816
3	6.4464	0.3668
4	6.2419	0.3746
5	4.9064	0.3539
6	5.8359	0.3816

n	Mean	Std. Dev.	Std. Err.	Exact
50	0.3697	0.0125	0.0018	0.3663

Level	z*	Lower	Upper	Covered
95%	1.96	0.3662	0.3732	TRUE
90%	1.64	0.3668	0.3726	FALSE
99%	2.58	0.3651	0.3743	TRUE

Figure 4:

Recalculating the simulated values in Excel multiple times, the 95% confidence interval rarely does not include the true value of 0.3663. Meanwhile, the 99% interval nearly always includes the true value, while the 90% interval fails to cover the true value far more often.

Problem 17

Because the demand can only take integer values, R's `sample` function is used to simulate daily demand for each produce item.

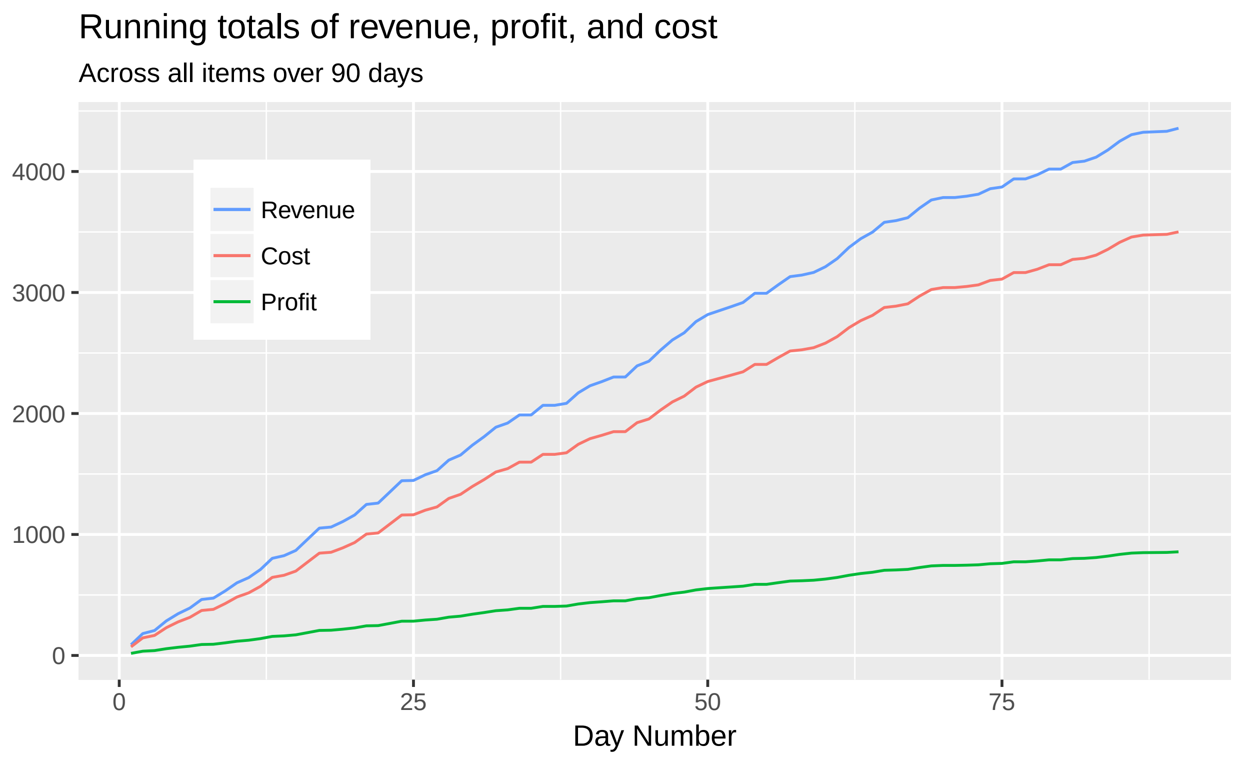
```r
library(tidyverse)

# set up provided data
produce <- c('oats', 'peas', 'beans', 'barley')
wholesale <- c(1.05, 3.17, 1.99, 0.95)
retail <- c(1.29, 3.76, 2.23, 1.65)
min_sales <- rep(0, 4)
max_sales <- c(10, 8, 14, 11)
num_days <- 90

# create container list for simulation results
walther <- list()

# create simulation for each item
for (p in 1:length(produce)) {
  set.seed(42) # set random number generator seed for replicability
  sales <- sample(min_sales[p]:max_sales[p], num_days, replace = TRUE)
  # store day number and calculate revenue, cost, and profit
  df <- data.frame(day_no = 1:num_days, sales) %>%
    mutate(revenue = sales * retail[p],
           cost = sales * wholesale[p],
           profit = revenue - cost)
  df$item <- rep(produce[p], num_days) # add item name
  # store in list
  walther[[p]] <- df
}

# collapse list into single data frame (tbl_df)
walther <- tbl_df(bind_rows(walther))

# calculate daily and running sums
daily <- walther %>%
  group_by(day_no) %>%
  summarise(revenue = sum(revenue),
            cost = sum(cost),
            profit = sum(profit)) %>%
  mutate(
    running_revenue = cumsum(revenue),
    running_cost = cumsum(cost),
    running_profit = cumsum(profit)
  )

# calculate total revenue, cost, and profit
tot <- daily[nrow(daily), 5:7]

# get daily and average sales by item
item <- walther %>%
  group_by(item) %>%
  mutate(avg_sales = cummean(sales))
```

Results

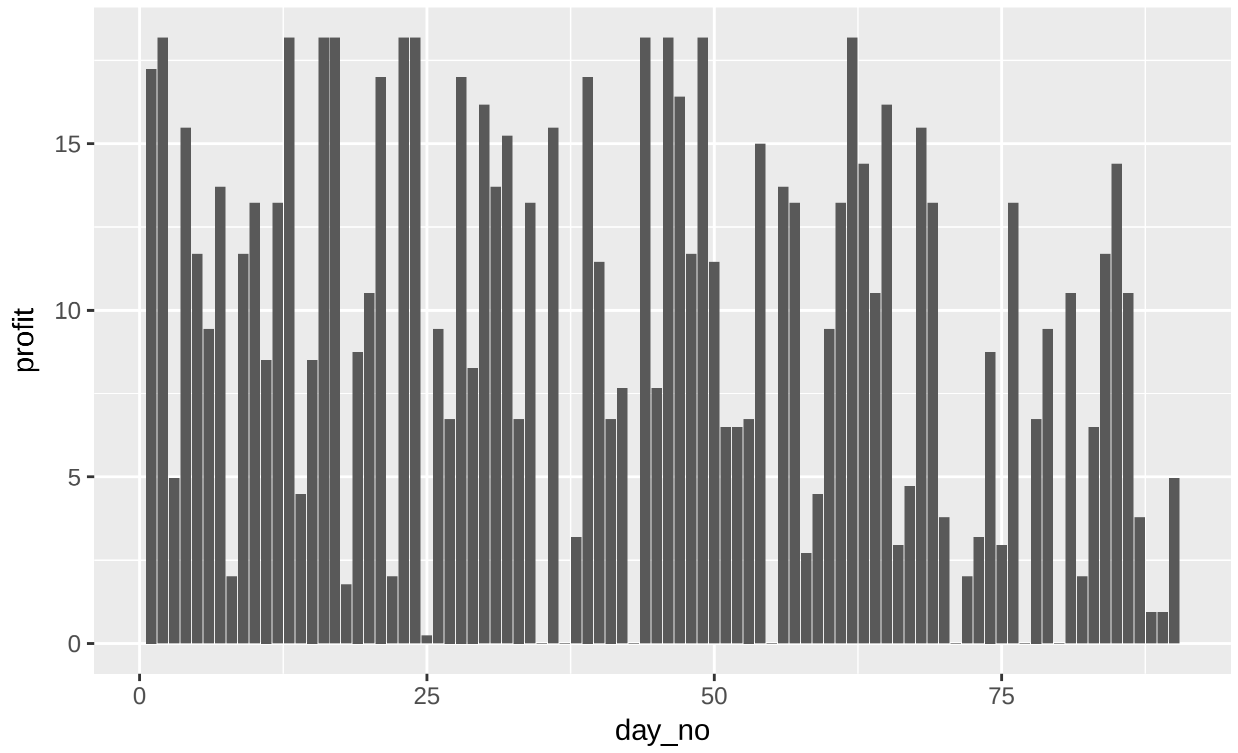
Following this simulation, the final total figures are as follows:

- Revenue: $4357.36
- Cost: $3500.76
- Profilt: $856.60

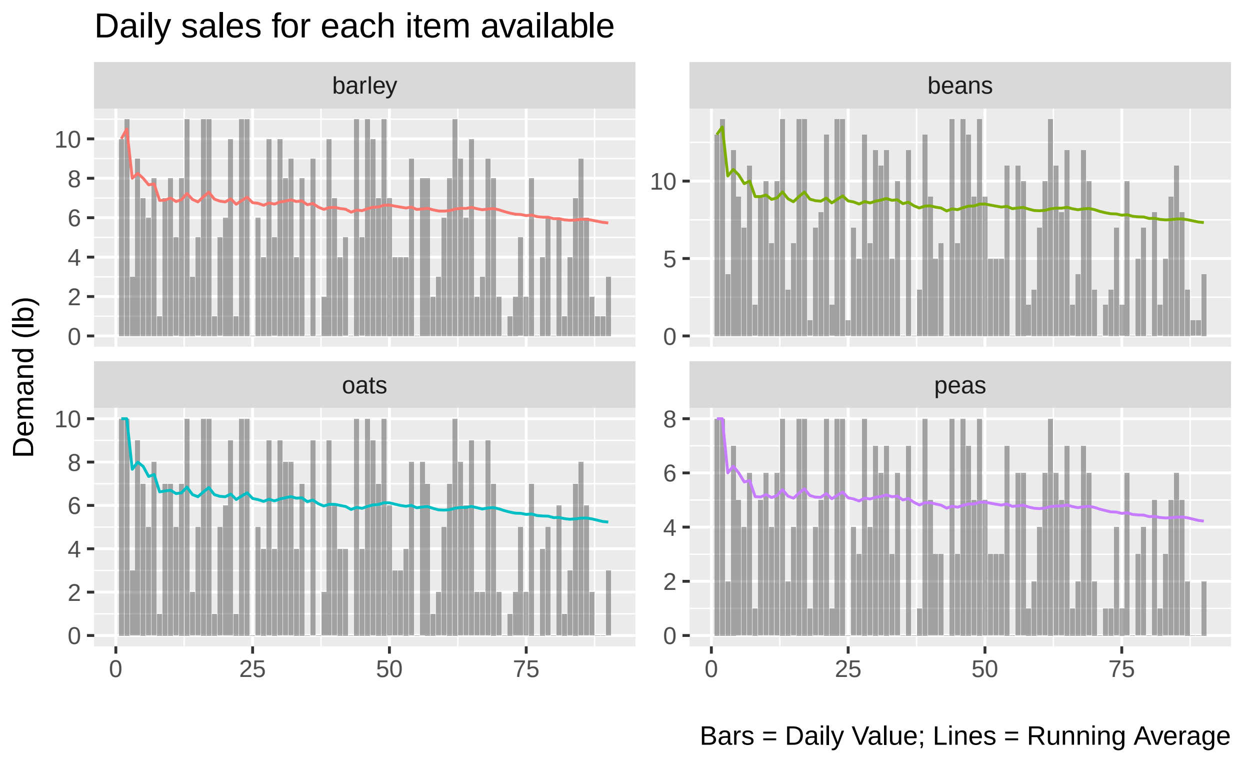
The progression of these values over time is presented below:



The profit day-by-day is shown below:



The daily demand and running average of daily demand are shown for each product:


Daily sales for each item available
barley
beans
oats
peas
Demand (lb)
Bars = Daily Value; Lines = Running Average