# Remedial Math 111 Group Project

James Lounds

December 5, 2021

# 1 Task 1

## 1.1 Brief

The Royal Mint wants to perform an investigation on the optimal production of each denomination of coin.
I need to:

- calculate the change given for each transaction under £1,
- calculate the distribution of the number of coins given as change,
- calculate the consequences of removing the 20p coin from circulation,
- discuss the consequences of removing the 1p and 2p coins from circulation.

## 1.2 Solution

I needed to define a few functions in MATLAB to make these calculations possible.

- coins_counts

  has arguments

  - prices - vector of prices to count change for
  - coins - the vector of the coins we have (unlimited) access to

  returns

  - counts - a matrix with columns corresponding to how many of each type of coin are in the optimal change give.

- coin_occurence_table

  has arguments

  - prices - vector of prices to count change for
  - coins - the vector of the coins we have (unlimited) access to

  returns

  - table - a table with the number of times a coin was used in the optimal change for each price

- coin_number_distribution_table

  has arguments

  - prices - vector of prices to count change for
  - coins - the vector of the coins we have (unlimited) access to

  returns

  - table - a table with the expected value of each denomination being used in the optimal solutioin of a price in prices

I then used a helpful tool developed by Víctor Martínez to convert the tables to LaTeX

## 1.3 Results

For the standard set of coins $\{50p, 20p, 10p, 5p, 2p, 1p\}$ and prices $\{0, 1, 2, \ldots, 99\}$

| Denomination | 0 | 1 | 2 |
|---|---|---|---|
| 50 | 50 | 50 | 0 |
| 20 | 40 | 40 | 20 |
| 10 | 60 | 40 | 0 |
| 5 | 50 | 50 | 0 |
| 2 | 40 | 40 | 20 |
| 1 | 60 | 40 | 0 |

| Occurences | Probability (%) |
|---|---|
| 0 | 1 |
| 1 | 6 |
| 2 | 17 |
| 3 | 28 |
| 4 | 28 |
| 5 | 16 |
| 6 | 4 |

For the set of coins without 1p and 2p $\{50p, 20p, 10p, 5p\}$ and prices $\{0, 5, 10, \ldots, 95\}$

| Denomination | 0 | 1 | 2 |
|---|---|---|---|
| 50 | 20 | 0 | 0 |
| 20 | 20 | 0 | 0 |
| 10 | 10 | 10 | 0 |
| 5 | 10 | 10 | 0 |

| Occurences | Probability (%) |
|---|---|
| 0 | 25 |
| 1 | 50 |
| 2 | 25 |

## 1.4 Conclusions

Our current coin system is reasonably well designed to pay for any natural value of pennies. If we were to limit transactions to multiples of five pence, we would be able to substantially reduce the number of coins needed to pay. However, this is not without consequence. There is a significant psychological aspect to pricing. Many shops will price things one penny under a whole price. This is purported to reduce the percieved cost in the customers' minds. If we were to force more quantized pricing, this psychological aspect may be reduced. Nonetheless, given the increase in purchases via non-cash methods, it would not be infeasible for retailers to round up or down to the nearest 5 pence. Cash is not free to handle, and can be more expesive than the fees cards processors charge. By reducing the amount of small change, the average value of the cash increases, thus decreasing the relative cost of handling cash.

## 1.5 Code

### 1.5.1 coin_occurences_table.m

```
1  function t = coin_occurences_table(prices, coins)
2      counts = coins_counts(prices, coins);
3      sz = [size(coins, 2) 4];
4      varNames = ["Denomination", "0", "1", "2"];
5      varTypes = ["double", "double", "double", "double"];
6      t = table('Size', sz, 'VariableTypes', varTypes, 'VariableNames', varNames);
7      t(:, 1) = table(reshape(coins, [size(coins, 2) 1]));
8
9      for i = 1:3
10         t(:, i + 1) = table(sum(counts + 1 == i, 2));
11     end
12 end
```

This function produces a table ofthe number of times each coin is used over the 99 transactions. Lines 2-6 initialise the table, while the remaining lines iterate up to the (hardcoded) maximum number of times a coin can be used optimally.

### 1.5.2 coin_number_distribution_table.m

```
1  function t = coin_number_distribution_table(prices, coins)
2      counts = coins_counts(prices, coins);
3      sz = [7 2];
4      varNames = ["Occurences", "Probability (%)"];
5      varTypes = ["double", "double"];
6      t = table('Size', sz, 'VariableTypes', varTypes, 'VariableNames', varNames);
7      t(:, 1) = table([0; 1; 2; 3; 4; 5; 6]);
8
9      for i = 0:6
10         t(i + 1, 2) = table(sum(sum(counts, 1) == i, 2) / (size(prices, 2) / 100));
11     end
12 end
```

This function is very similar to the previous, but instead calculates the expected value of the number of each coin used in some optimal change.

### 1.5.3 coins_counts

```
function counts = coins_counts(prices, coins)
    % preallocation
    counts = zeros(size(coins, 2), size(prices, 2));

    for i = 1:size(coins, 2)
        coinValue = coins(i);
        n = fix(prices / coinValue);
        counts(i, :) = n;
        prices = prices - (n * coinValue);
    end

end
```

This function calculates a matrix with each entry being the number of coins of the denomination (indicated by the row) needed for the optimal change for the price (indicated by the column). It does this by iterating through each price, and appending the correct coin count vector to the end of the counts matrix. Each entry in the column vector is calculated by floor division, then leaving the remainder for the next iteration.

### 1.5.4 task_1.m

```
sm_purchases = 0:99;
coins = [50 20 10 5 2 1];

table2latex(coin_occurences_table(sm_purchases, coins), '../tables/coin_occurence_normal.tex')
table2latex(coin_number_distribution_table(sm_purchases, coins), '../tables/coin_distribution_normal.tex')

prices = 5 * 0:19;
coins = [50 20 10 5];
table2latex(coin_occurences_table(prices, coins), '../tables/coin_occurence_no_2_1.tex')
table2latex(coin_number_distribution_table(prices, coins), '../tables/coin_distribution_no_2_1.tex')
```

This code only invokes the previously defined functions for our paramterers and puts the tables in TeX files.

# 2 Task 2

## 2.1 Brief

The price of gold bullion can be modelled using Random Walk Theory. If Party A has £6 and Party B has £14, and they were to play a game where a fair coin was flipped, and the winner gave the loser £1, this would produce a random walk. I need to calculate:

- The probability Part A wins.
- The probability that Party A loses given that they were within £1 of winning.

## 2.2 Solution

I defined only one function in my solution:

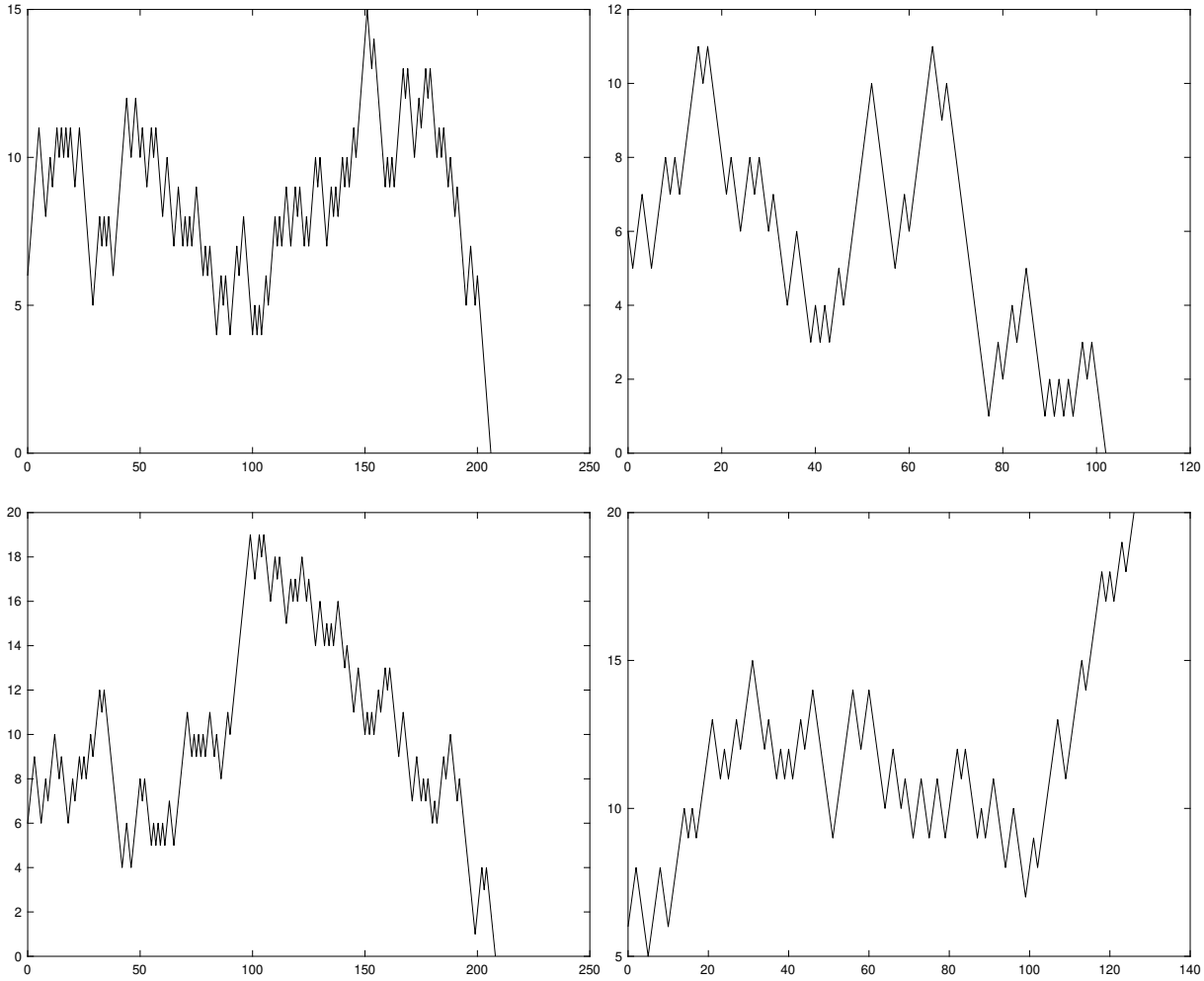
- play_game
  has arguments
  - initial_scores - the amount of money A and B have at the start, as a row vector.

  returns
  - scores - a matrix where each row is the amount of money A and B have after a round.

## 2.3 Results

| Probability A Wins | Probability A Loses Closely |
|---|---|
| 0.2978 | 0.015 |

Note: only the bottom two plots have scale 0-20. Teh bottom left plot is a close loss for A



## 2.4 Code

### 2.4.1 play_game.m

```
function scores = play_game(initialScores)
    scores = initialScores;

    while sum(scores ⩽ 0) == 0
        flip = 2 * randi([0 1], 1) - 1;
        scores(end + 1, :) = scores(end, :) + flip * [1 -1];
    end

end
```

This code loops until either player has a score of 0. On each iteration, it generates a flip $\in \{-1, 1\}$. It then adds a new row to the end of the scores matrix, with each playeer's score incremented or decremented depending on the value of flip.

### 2.4.2 task_2.m

```
a_wins = 0;
a_close_losses = 0;
scores = [6 14];

n = 5000;

for i = 1:n
    result = play_game(scores);

    if result(end, 1) == 20
        a_wins = a_wins + 1;
    elseif sum(result(:, 1) == 19) ~= 0
        a_close_losses = a_close_losses + 1;

    end

end

results = table(a_wins / n, a_close_losses / n);
results.Properties.VariableNames = {'Probability A Wins' 'Probability A Loses Closely'};
table2latex(results, '../tables/coin_game.tex')

result = play_game(scores);
a_scores = result(:, 1);
p = plot(0:size(a_scores) -1, a_scores);
print -deps epsFig
```

We start by initialising some variables. We then play $n$ random games and record if A won, or if A closely lost. This data is stored in a table then exported to LaTeX. We then run one more random game, which is plotted, and then exported to LaTeX