An Analysis of Optimal Soil Choice

Contract Consultants: Matthew Knorr, Nels Blair, and Sandy Auttelet

August 30, 2024

Executive Summary

Business Parameters

From the initial parameters, our reference numbers are that one meter of bamboo weighs about 10kg and each kg is worth 12 dollars. We can begin selling bamboo once it has grown up to 3 meters. These numbers were used in the calculations throughout this report and can be changed if the market shifts.

Growth Data

These plots compare the growth of plants in x-grow brand fertilizer and y-grow brand fertilizer. The third plot containing both sets of data marks where plants are eligible to be sold.

If you choose to grow with x fertilizer, your plants will grow faster in the beginning of a 60 day cycle, but the growth rate will begin to decline below the growth rate of y at about 26 days. The plants grown in x or y will come to the same average growth rate and thus same height at 53 days. After 53 days, plants grown with y fertilizer have a higher growth rate for the rest of the duration of the trial experiment. We conclude that there are significant differences in bamboo growth when using the fertilizers from brands x-grow versus y-grow.

Growth Rates

Table 1: Comparing Fertilizers

Characteristic	Fertilizer x	Fertilizer y
Days to 3m	12.7	25.42
Rate of Growth at 3m	0.46	0.232
Optimal Growth Cycle (days)	20.3	48.81
Optimal Choice	Yes	No

The plants grown in x-grow brand fertilizer are eligible for harvest on the 18 hours into the 13th day after planting. Plants grown in y-grow brand fertilizer are eligible for harvest 13 hours into the 26th day after planting. When the plants reach a height of 3 meters, the growth rates of bamboo with fertilizers x and y are, respectively, 0.46 and 0.232 meters per day.

Generally, it is sub-optimal to harvest a plant which is in the middle of a relatively high-growth period, but may be optimal to harvest directly after such a period. The optimal growth cycles are 20.3 and 48.81 days for x and y, respectively. These cycles will yield a profit of 730 and 1,030 dollars for fertilizer x and y, respectively. It is important to note that these cycles are of different duration. For direct comparison, x yields 34 dollars per day and y yields 21 dollars per day on average in an optimal cycle.

1 Decision

Characteristics	x-grow Fertilizer	y-grow Fertilizer
Cost per Plant	1 dollar	1.25 dollar
Profit per Plant in Optimal Cycle of Growth	690 dollars	1,030 dollars

Based on our analysis of the data, we have concluded that x-grow brand fertilizer is the optimal choice if you can harvest repeatedly in 60 days. The cost per plant of fertilizer y (1.25) is negligible compared with the revenue earned from the same plant (1,200). So, we simply disregard this cost of fertilizer without loss of any margin of comparison. The same applies for the cost per plant of fertilizer x (1.00), which in three (optimal) cycles becomes 3 compared with the revenue earned from the same individual physical growth slot (1950). Even though at the end of the full 60 day trial, y is favored, when we include factors like cost of fertilizer (negligible), and length of optimal growing cycle (significantly shorter for x) the clear winner, yielding more dollars per day is fertilizer x.

2 Profits of Scale

We assume the company operates at full capacity to produce 20,000 plants in the 60 day time frame, and has the corresponding turnover of product. We seek to use profit per plant, then scale up to 20,000 plants to get a number that represents profits per 60 days at full capacity.
The one dollar or one dollar twenty-five cents cost is actually quite negligible when compared with the revenue generated by each shoot cycle ($1950and1200$ for x and y respectively). Therefore, we effectively disregard the cost of each brand of fertilizer and focus on the difference in revenue generated by each optimal cycle.
The output of the above code indicates that fertilizers x and y will grow plants that generate revenue of 39 million and 24 million respectively. The 15 million difference between them is the difference of the total marginal benefit from employing fertilizer x as opposed to fertilizer y.

3 Appendix

```python
#Main File
import numpy as np
import pandas as pd
import Build_data_report as BDR
import Plot_data_figures as PDF


#User input business perameters
x_cost = 1.0
y_cost = 1.25
dollar_per_kg = 12
kg_per_m = 10
dollar_per_m = dollar_per_kg*kg_per_m
num_plants = 20000

#x_data = np.loadtxt("x_grow.csv", delimiter=",")
#y_data = np.loadtxt("y_grow.csv", delimiter=",")
x_data = pd.read_csv("C:/Users/sandy/OneDrive/Documents/Summer 2023/Computing - 300/Final/x_grow.csv", delimiter=",")
y_data = pd.read_csv("C:/Users/sandy/OneDrive/Documents/Summer 2023/Computing - 300/Final/y_grow.csv", delimiter=",")
x_data = np.array(x_data)
y_data = np.array(y_data)

x_data_list_0,x_data_list_1 = BDR.build_data_list(x_data)
y_data_list_0,y_data_list_1 = BDR.build_data_list(y_data)

sell_day_x = BDR.find_sell_day(x_data)
sell_day_y = BDR.find_sell_day(y_data)
sell_mom_x = BDR.find_sell_moment(x_data,sell_day_x)
sell_mom_y = BDR.find_sell_moment(y_data,sell_day_y)

x_growth_rate = BDR.find_growth_rate(x_data, sell_day_x)
y_growth_rate = BDR.find_growth_rate(y_data, sell_day_y)
x_growth_rates = BDR.find_growth_rates(x_data)
y_growth_rates = BDR.find_growth_rates(y_data)
x_max_rate, x_max_rate_day = BDR.find_peak_growth(x_data, x_growth_rates)
y_max_rate, y_max_rate_day = BDR.find_peak_growth(y_data, y_growth_rates)

x_optimal_cycle, x_max_profit_per_day = BDR.find_optimal_duration\
    (x_data,x_cost,dollar_per_m,sell_day_x)
y_optimal_cycle, y_max_profit_per_day = BDR.find_optimal_duration\
    (y_data,y_cost,dollar_per_m,sell_day_y)


profit_data_x, cycles_x = BDR.find_profit_data(x_data,sell_day_x, dollar_per_m,x_cost)
profit_data_y, cycles_y = BDR.find_profit_data(y_data,sell_day_y, dollar_per_m,y_cost)
```

```
    x_max_profit, x_optimal_cycles = BDR.find_actual_profit(x_data, \
        sell_day_x, dollar_per_m, x_cost)
    y_max_profit, y_optimal_cycles = BDR.find_actual_profit(y_data, \
        sell_day_y, dollar_per_m, y_cost)
    x_total_profit = x_max_profit*num_plants
    y_total_profit = y_max_profit*num_plants

    if x_total_profit > y_total_profit:
        total_saved_money = round((x_total_profit - y_total_profit),2)
    if y_total_profit > x_total_profit:
        total_saved_money = round((y_total_profit - x_total_profit),2)
    if y_total_profit == x_total_profit:
        total_saved_money = 0

    x_profit_data,x_cycles = BDR.find_profit_data(x_data,sell_day_x,dollar_per_m,x_cost)
    y_profit_data,y_cycles = BDR.find_profit_data(y_data,sell_day_y,dollar_per_m,y_cost)

    report_x_profit = BDR.report_million(BDR.find_mantissa(x_total_profit))
    report_y_profit = BDR.report_million(BDR.find_mantissa(y_total_profit))
    report_total_profit = BDR.report_million(BDR.find_mantissa(total_saved_money))

    #Plotting data report:

    #Plots curve fit for data
    PDF.plot_curve_fit_x(x_data_list_0,x_data_list_1)
    PDF.plot_curve_fit_y(y_data_list_0,y_data_list_1)

    #Plots data as line segment seperately
    PDF.plot_x_data(x_data_list_0,x_data_list_1)
    PDF.plot_y_data(y_data_list_0,y_data_list_1)

    #Plots both sets together
    PDF.plot_x_y_data(x_data_list_0,x_data_list_1,y_data_list_0,y_data_list_1)
    #Plots together with 3m mark
    PDF.plot_sell_day(x_data_list_0,x_data_list_1,y_data_list_0,y_data_list_1)

    #Plots growth rates by day
    PDF.plot_growth_rate(x_data_list_0,x_data_list_1,y_data_list_0,y_data_list_1,x_growth_rates,y_growth_rates)

    #Plots growth per cycle
    PDF.make_barchart_x(x_profit_data,x_cycles)
    PDF.make_barchart_y(y_profit_data,y_cycles)


    #Printing report set up for x and y grow brand fertilizer
```

```
    #Prints sell day information
    print(f'We can sell bamboo shoots grown with x fertilizer after \
    {int(sell_day_x)} days and roughly {int(sell_mom_x)} hours after planting.')
    print(f'\nWe can sell bamboo shoots grown with y fertilizer after \
    {int(sell_day_y)} days and roughly {int(sell_mom_y)} hours after planting.')

    #Prints groth rates
    print(f'\n\nThe growth rate of bamboo grown with x fertilizer at \
    point of sell is {BDR.find_mantissa(x_growth_rate)} meters per day.')
    print(f'\nThe growth rate of bamboo grown with y fertilizer at \
    point of sell is {BDR.find_mantissa(y_growth_rate)} meters per day.')

    #Printing max rates and analysis of optimal cycles
    print(f'\n\nFertilizer x results in a peak growth rate of {BDR.find_mantissa(x_max_rate)} meters per day on day {round(x_max_rate_day)}')
    print(f'\nFertilizer y results in a peak growth rate of {BDR.find_mantissa(y_max_rate)} meters per day on day {round(y_max_rate_day)}.')

    #Paragraph specific to this data and visual analysis. Not valid for new data.
    print('\nIf you choose to grow with x fertilizer, your plants will \
    grow faster in the beginning of a 60 day cycle, but the growth \
    rate will begin to decline at about 26 days. After about 30 days, \
    both sets will be growing at roughly the same rate. At this \
    point, plants grown with y fertilizer will begin to grow faster. \
    Both plant sets will come to the same average growth rate and \
    thus same height at 53 days, after which y fertilizer grown \
    plants have a faster average growth rate.')

    #Prints optimal cycle days and max profit in that cycle
    print(f'\n\nThe maximum profit per {BDR.find_mantissa(x_optimal_cycle)} days, the optimal cycle, per plant \
    grown in x fertilizer will be {BDR.find_mantissa(x_max_profit_per_day*x_optimal_cycle)} dollars.')

    print(f'\nThe maximum profit per {BDR.find_mantissa(y_optimal_cycle)} days, the optimal cycle, per plant \
    grown in y fertilizer will be {BDR.find_mantissa(y_max_profit_per_day*y_optimal_cycle)} dollars.')

    #Prints the optimal choice
    print(f'\n\nThe maximum profit per {BDR.find_mantissa(y_optimal_cycle)} \
    days, the optimal cycle, per plant grown in y fertilizer will be \
    {BDR.find_mantissa(y_max_profit_per_day*y_optimal_cycle)} dollars.')
    if x_max_profit_per_day > y_max_profit_per_day:
        print("\nOptimal fertilizer choice is x because it yields more \
```

```
    dollars per day and in the long run pays off.")
    if y_max_profit_per_day > x_max_profit_per_day:
        print("\nOptimal fertilizer choice is y because it yields more \
    dollars per day and in the long run pays off.")
    if x_max_profit_per_day == y_max_profit_per_day:
        print("\nBoth fertilizers are equavalent in optimal profit.")

    #Notes irrelevent cost
    print('\nNote that the cost per plant is negligible because the profit is roughly one thousand times larger than the cost.')

    #Prints final data analysis
    print(f'\n\nAfter {len(x_data)} days as {int(BDR.find_mantissa(x_optimal_cycles))} \
    cycle(s) of {int(BDR.find_mantissa(len(x_data)/x_optimal_cycles))} days in \
    fertilizer x, the profit per plant is {BDR.find_mantissa(x_max_profit)}. \
    Therefore, the total profit for all {num_plants} plants at a time \
    is {report_x_profit}.')

    print(f'\nAfter {len(y_data)} days as {int(BDR.find_mantissa(y_optimal_cycles))} \
    cycle(s) of {int(BDR.find_mantissa(len(y_data)/y_optimal_cycles))} days in \
    fertilizer y, the profit per plant is {BDR.find_mantissa(y_max_profit)}. \
    Therefore, the total profit for all {num_plants} plants at a time \
    is {report_y_profit}.')

    print(f'\nFinally, the total profit saved from using the optimal \
    fertilizer is {report_total_profit}')
```

```

#Build_data_report File for functions
import numpy as np
import math

#For rounding later on:
def find_mantissa(num):
    scale = int(round(np.log10(num),0))-3
    mantissa = int(round(num/10**scale,0))
    rounded_num = mantissa*10**scale
    rounded_num = round(rounded_num,3)
    return rounded_num

def report_million(num):
    if num >= 1000000:
        num = int(num/1000000)
        return f'{num} Million'
    if num < 1000000:
        return num

#Build data report
def build_data_list(data):
    data_list_0 = []
    data_list_1 = []
    for i in range(len(data)):
        data_list_0.append(data[i][0])
        data_list_1.append(data[i][1])
    return data_list_0,data_list_1

#Finds day eligible for selling
def find_sell_day(data):
    for i in range(len(data)):
        if data[i][1] >= 3.0:
            max_day = data[i][0]
            min_day = data[i-1][0]
            grow_max = data[i][1]
            grow_min = data[i-1][1]
            mom_at_3 = ((grow_max-3)*max_day+(3-grow_min)*min_day)/(
grow_max-grow_min)
            return mom_at_3
    raise Exception("Did not reach 3 meters in observed duration.
Expand duration or adjust expected height.")

def find_sell_moment(data,sell_day):
    extra = sell_day - int(sell_day)
    extra = 24*extra
    extra = round(extra,0)
    return extra

def find_growth_rate(data, sell_day):
    for i in range(len(data)):
        if data[i][0] >= sell_day:
            dh = data[i+1][1] - data[i-1][1]
```

```python
                dt = data[i+1][0] - data[i-1][0]
                rate = dh/dt
                return rate

    def find_growth_rates(data):
        growth_rates = []
        dh0 = data[1][1] - data[0][1]
        dt0 = data[1][0] - data[0][0]
        growth_rates.append(dh0/dt0)
        dh1 = data[2][1] - data[0][1]
        dt1 = data[2][0] - data[0][0]
        growth_rates.append(dh1/dt1)
        dh2 = data[4][1] - data[0][1]
        dt2 = data[4][0] - data[0][0]
        growth_rates.append(dh2/dt2)
        for i in range(3,len(data)-3):
            dh = data[i+3][1] - data[i-3][1]
            dt = data[i+3][0] - data[i-3][0]
            growth_rates.append(dh/dt)
        dhe = data[len(data)-1][1] - data[len(data)-5][1]
        dte = data[len(data)-1][0] - data[len(data)-5][0]
        growth_rates.append(dhe/dte)
        dhe = data[len(data)-1][1] - data[len(data)-3][1]
        dte = data[len(data)-1][0] - data[len(data)-3][0]
        growth_rates.append(dhe/dte)
        dhe = data[len(data)-1][1] - data[len(data)-2][1]
        dte = data[len(data)-1][0] - data[len(data)-2][0]
        growth_rates.append(dhe/dte)
        return growth_rates

    def find_peak_growth(data,growth_rates):
        new_max_rate = 0
        for i in range(len(growth_rates)):
            max_rate = new_max_rate
            if growth_rates[i] > max_rate:
                max_rate = growth_rates[i]
                max_day = data[i][0]
            new_max_rate = max_rate
        return max_rate, max_day

    def find_optimal_duration(data,cost,dollar_per_m,sell_day):
        t_max = len(data)
        profit = []
        max_day = 0
        new_max = 0
        for i in range(int(round(sell_day,0)),t_max):
            max_profit = new_max
            profit.append((data[i][1]*dollar_per_m-cost)/data[i][0])
            new_max = max(profit)
            if new_max > max_profit:
                max_day = data[i][0]
        return max_day,max_profit

    def find_actual_profit(data,sell_day,dollar_per_m,cost):
```

```
        cycles = int(math.trunc(len(data)/int(round(sell_day,0))))
        max_profit = 0
        for i in range(1,cycles+1):
            days = int(math.trunc(len(data)/i))
            height = data[days-1][1]
            profit = height*dollar_per_m - cost
            if profit*i >= max_profit:
                max_profit = profit*i
                optimal_cycles = i
        return max_profit, optimal_cycles

    #For building a histogram
    def find_actual_profit_cycles(data,sell_day,dollar_per_m,cost,days):
        if sell_day > days:
            raise Exception("The requested duration is shorter than the eligible sell day. Choose a longer duration or different fertilizer.")
        cycles = int(math.trunc(len(data)/int(round(days,0))))
        height = data[int(round(days,0))-1][1]
        profit = cycles*(height*dollar_per_m - cost)
        return find_mantissa(profit)

    def find_profit_data(data,sell_day,dollar_per_m,cost):
        num = len(data)
        nums = [num]
        profits = []
        for i in range(2,len(data)):
            if num/i < sell_day:
                break
            nums.append(num/i)
        for i in range(len(nums)):
            profits.append(2*find_actual_profit_cycles(data,sell_day,dollar_per_m,cost,nums[i]))
        return profits, nums

```

```
#Plot_data_figures file for plotting
import numpy as np
import matplotlib.pyplot as plt

#Functions made for curve fit.
#Will need to change for different data.
#constants = sc.optimize.curve_fit(f_y,y_data_list_0,y_data_list_1)

def f_y(t,x0,x1,x2,x3):
    f = x0/(1+np.exp(-x1*t+x2))+x3
    return f

def f_x(t,x0,x1,x2,x3,x4,x5,x6,x7,x8):
    f = x0*t**2 + x1*t**3 + x2*t**4 + x3*t + x4 + x5*t**5 + x6*t**6 + x7*t**7 + x8*t**8
    return f

def plot_curve_fit_x(x_data_list_0,x_data_list_1):
    tx = np.linspace(min(x_data_list_0),max(x_data_list_0),1000)
    fig1 = plt.figure(1)
    fig1.suptitle("Measured and Fit Data for x-grow", fontsize=16)
    plt.plot(x_data_list_0,x_data_list_1)
    plt.xlabel('Days of Growth')
    plt.ylabel('Height (m)')
    plt.scatter(x_data_list_0,x_data_list_1,color='red')
    plt.plot(tx,f_x(tx,-6.51968382e-02,  1.05973801e-02, -6.73468343e-04,  2.59080135e-01,
                  3.55047909e-01,  2.16533793e-05, -3.77703269e-07,  3.41898145e-09,
                 -1.26162606e-11))
    labels = ('Measurements','Polynomial Curve Fit')
    plt.legend(labels)

def plot_curve_fit_y(y_data_list_0,y_data_list_1):
    ty = np.linspace(min(y_data_list_0),max(y_data_list_0),1000)
    fig2= plt.figure(2)
    fig2.suptitle("Measured and Fit Data for y-grow", fontsize=16)
    plt.xlabel('Days of Growth')
    plt.ylabel('Height (m)')
    plt.scatter(y_data_list_0,y_data_list_1,color='red')
    plt.plot(ty,f_y(ty,10.70286051,  0.09901738,  3.46901829,  0.09466854))
    labels = ('Measurements','Logistic Curve Fit')
    plt.legend(labels)

#Plot 1 of just x data
def plot_x_data(x_data_list_0,x_data_list_1):
    fig3 = plt.figure(3)
    fig3.suptitle("Growth with Fertilizer x", fontsize=16)
    plt.plot(x_data_list_0,x_data_list_1)
    plt.xlabel('Days of Growth')
    plt.ylabel('Height (m)')

```

```
    #Plot 2 of just y data
    def plot_y_data(y_data_list_0,y_data_list_1):
        fig4 = plt.figure(4)
        fig4.suptitle("Growth with Fertilizer y", fontsize=16)
        plt.plot(y_data_list_0,y_data_list_1)
        plt.xlabel('Days of Growth')
        plt.ylabel('Height (m)')

    #Plot 3 of x and y data
    def plot_x_y_data(x_data_list_0,x_data_list_1,y_data_list_0, y_data_list_1):
        fig5 = plt.figure(5)
        fig5.suptitle("Growth with Fertilizer x and y", fontsize=16)
        plt.plot(x_data_list_0,x_data_list_1)
        plt.plot(y_data_list_0,y_data_list_1)
        plt.xlabel('Days of Growth')
        plt.ylabel('Height (m)')
        labels = ('x Fertilizer','y Fertilizer')
        plt.legend(labels)

    #Plot 4 marks where 3m exists to find when we can sell
    def plot_sell_day(x_data_list_0,x_data_list_1,y_data_list_0, y_data_list_1):
        fig6 = plt.figure(6)
        fig6.suptitle("When We Can Sell", fontsize=16)
        plt.plot(x_data_list_0,x_data_list_1)
        plt.plot(y_data_list_0,y_data_list_1)
        plt.axhline(3,linestyle='--')
        plt.xlabel('Days of Growth')
        plt.ylabel('Height (m)')
        labels = ('x Fertilizer','y Fertilizer','3m Achievement')
        plt.legend(labels)

    #Plot 5 Growth Rate
    def plot_growth_rate(x_data_list_0,x_data_list_1,y_data_list_0, y_data_list_1,x_growth_rates,y_growth_rates):
        fig7 = plt.figure(7)
        fig7.suptitle("Growth Rates", fontsize=16)
        plt.plot(x_data_list_0,x_growth_rates)
        plt.plot(y_data_list_0,y_growth_rates)
        plt.xlabel('Days of Growth')
        plt.ylabel('Rate of Growth')
        labels = ('x Fertilizer','y Fertilizer')
        plt.legend(labels)

    def make_barchart_x(profit_data,cycles):
        fig8 = plt.figure(8)
        fig8.suptitle("60-Day Profit for Each Cycle Duration in x Fertilizer", fontsize=16)
        plt.bar(cycles,profit_data)
        plt.xlabel('Cycle Duration in Days')
        plt.ylabel('Profit per Plant in Ten Thousand Dollars')

    def make_barchart_y(profit_data,cycles):
```

```
        fig9 = plt.figure(9)
        fig9.suptitle("60-Day Profit for Each Cycle Duration in y Fertilizer", fontsize=16)
        plt.bar(cycles,profit_data)
        plt.xlabel('Cycle Duration in Days')
        plt.ylabel('Profit per Plant in Ten Thousand Dollars')

```