Pricing Case Study

You're launching a ride-hailing service that matches riders with drivers for trips between the Toledo Airport and Downtown Toledo. It'll be active for only 12 months. You've been forced to charge riders $30 for each ride. You can pay drivers what you choose for each individual ride.

The supply pool ("drivers") is very deep. When a ride is requested, a very large pool of drivers see a notification informing them of the request. They can choose whether or not to accept it. Based on a similar ride-hailing service in the same market, you have some data on which ride requests were accepted and which were not. (The PAY column is what drivers were offered and the ACCEPTED column reflects whether any driver accepted the ride request.)

The demand pool ("riders") can be acquired at a cost of $30 per rider at any time during the 12 months. There are 10,000 riders in Toledo, but you can't acquire more than 1,000 in a given month. You start with 0 riders. "Acquisition" means that the rider has downloaded the app and may request rides. Requested rides may or may not be accepted by a driver. In the first month that riders are active, they request rides based on a Poisson distribution where lambda = 1. For each subsequent month, riders request rides based on a Poisson distribution where lambda is the number of rides that they found a *match* for in the previous month. (As an example, a rider that requests 3 rides in month 1 and finds 2 matches has a lambda of 2 going into month 2.) If a rider finds no matches in a month (which may happen either because they request no rides in the first place based on the Poisson distribution or because they request rides and find no matches), they leave the service and never return.

Submit a written document that proposes a pricing strategy to maximize the profit of the business over the 12 months. You should expect that this singular document will serve as a proposal for

1. A quantitative executive team that wants to know how you're thinking about the problem and what assumptions you're making but that does not know probability theory
2. Your data science peers so they can push on your thinking

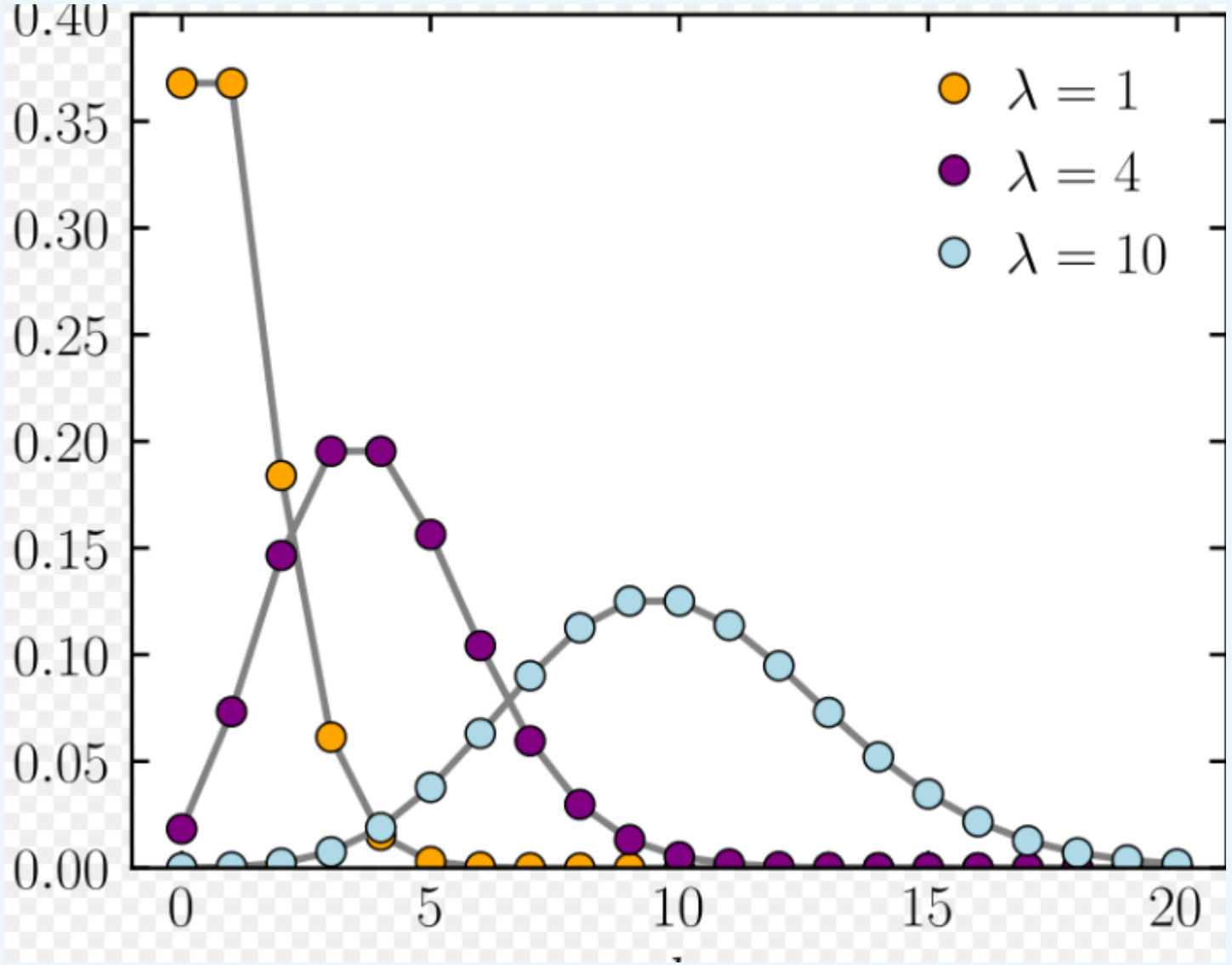
Car Pooling – Price Stategy

A Case Study Proposal for HealthClip

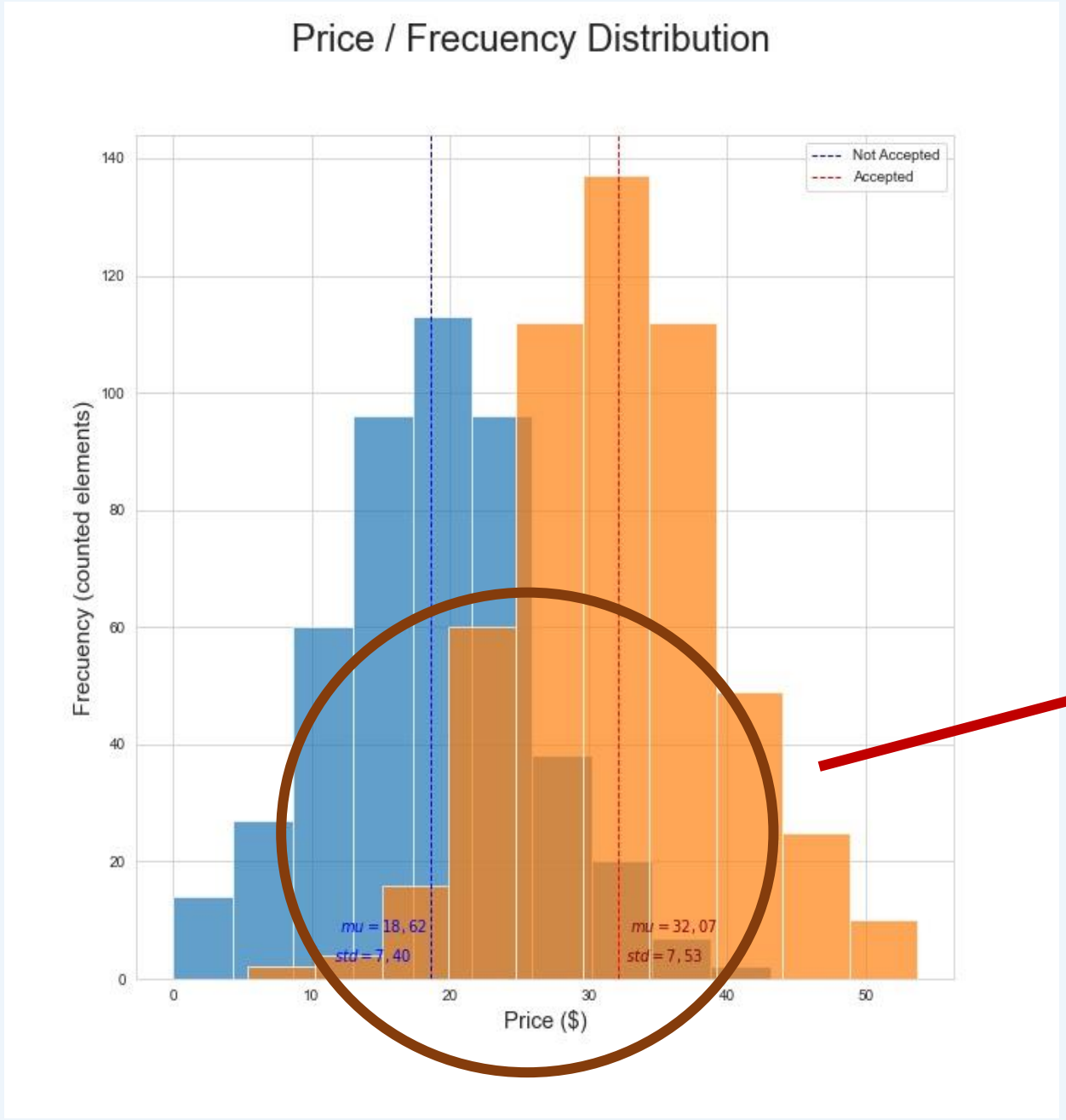
Pablo Cumbrera Conde

December, 05 2022

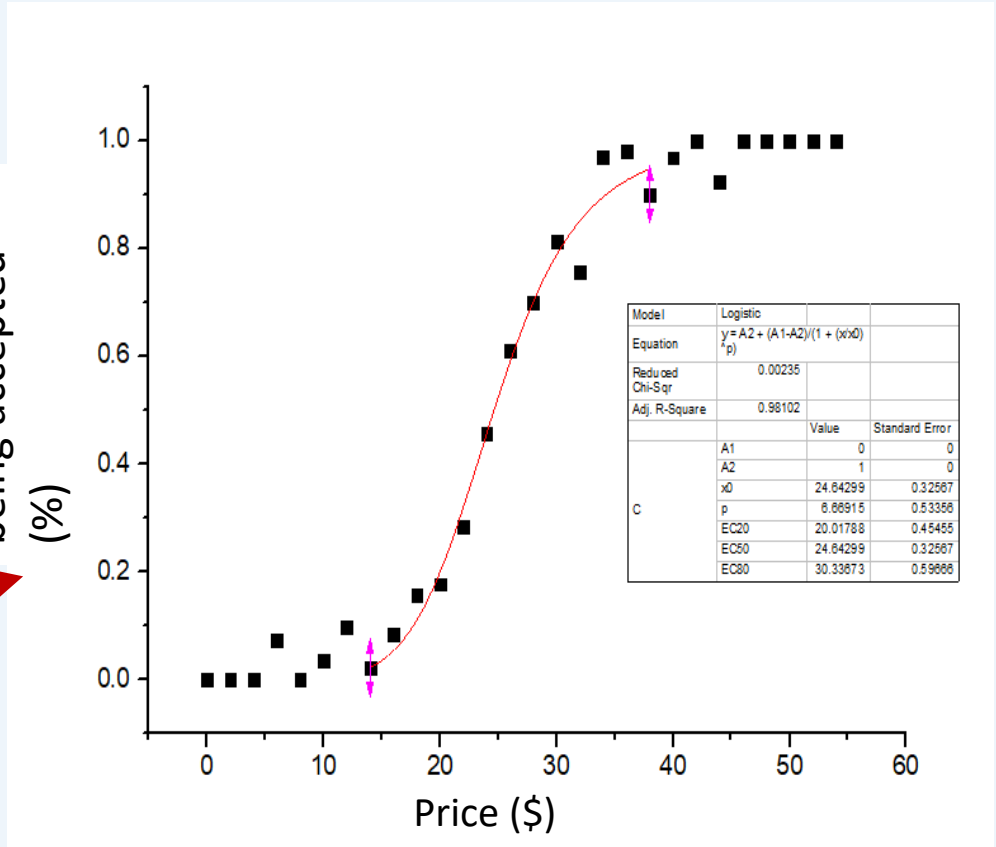
Case Key distribution: Poisson Distribution



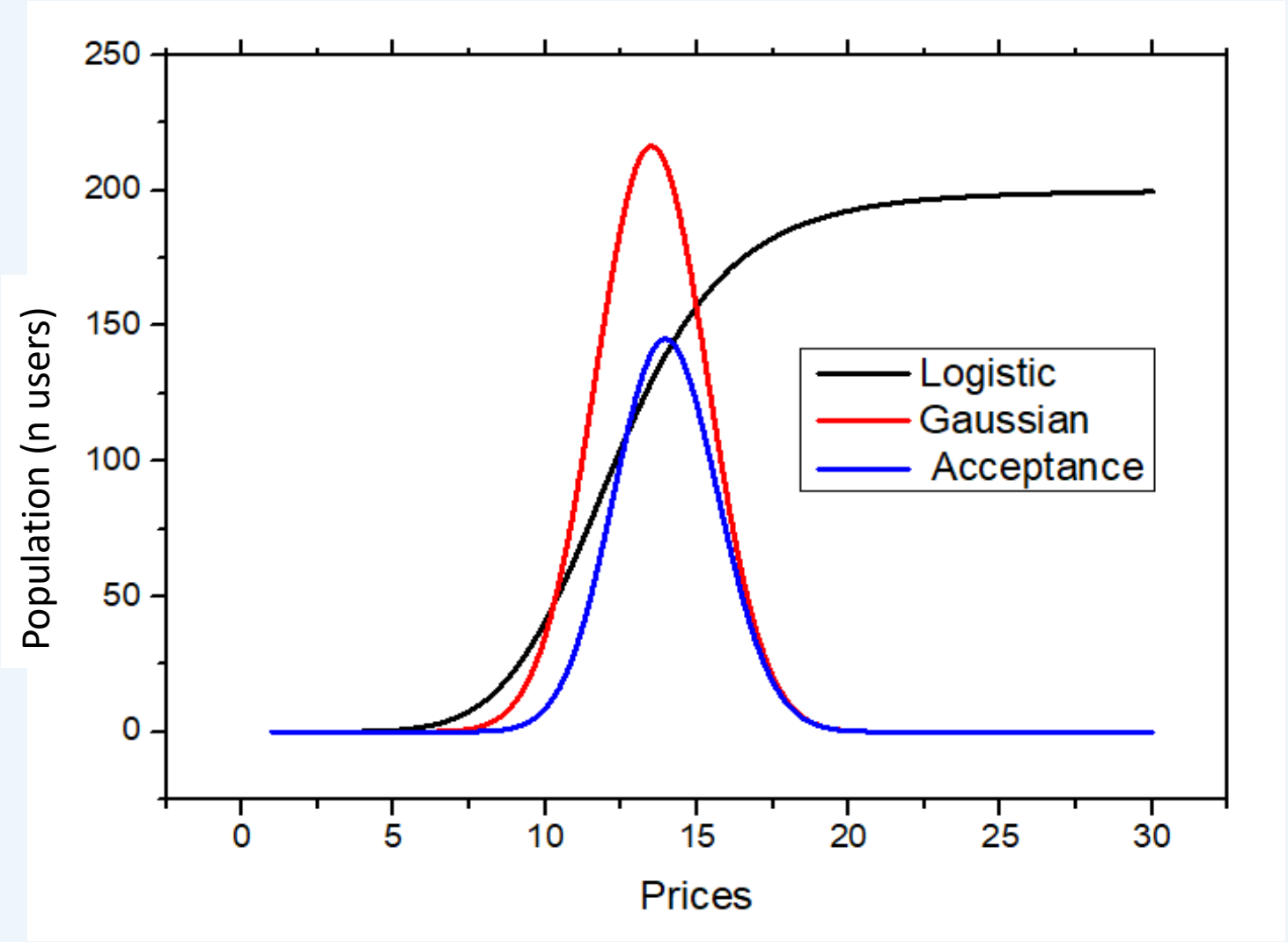
Poisson distribution examples according to lambda.



Study based on other company data.

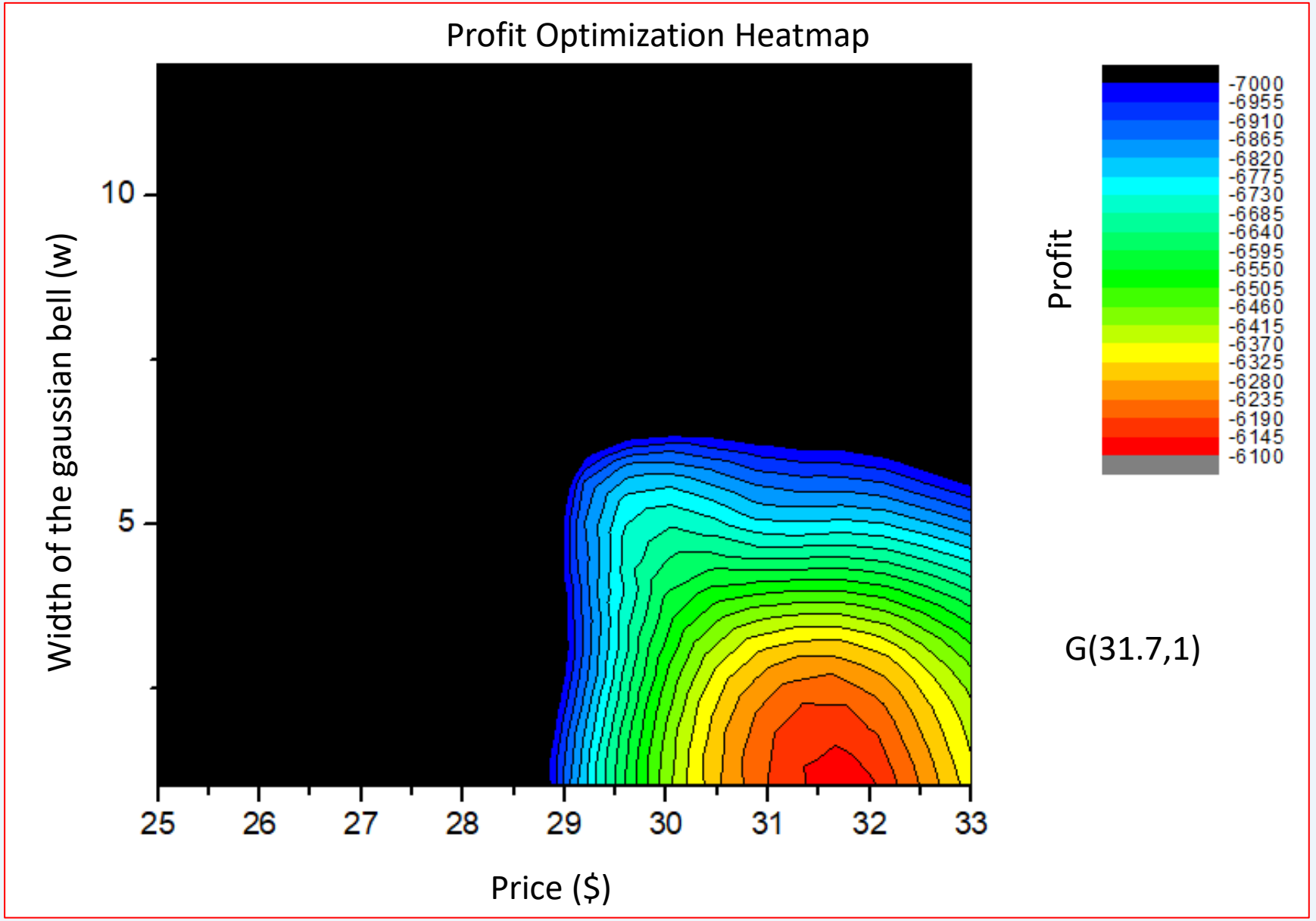


Procedure explanation



Logistic (Acceptance ratio) * Gaussian (Distribution) = Acceptance distribution

How to calculate the optimal Price / Profit

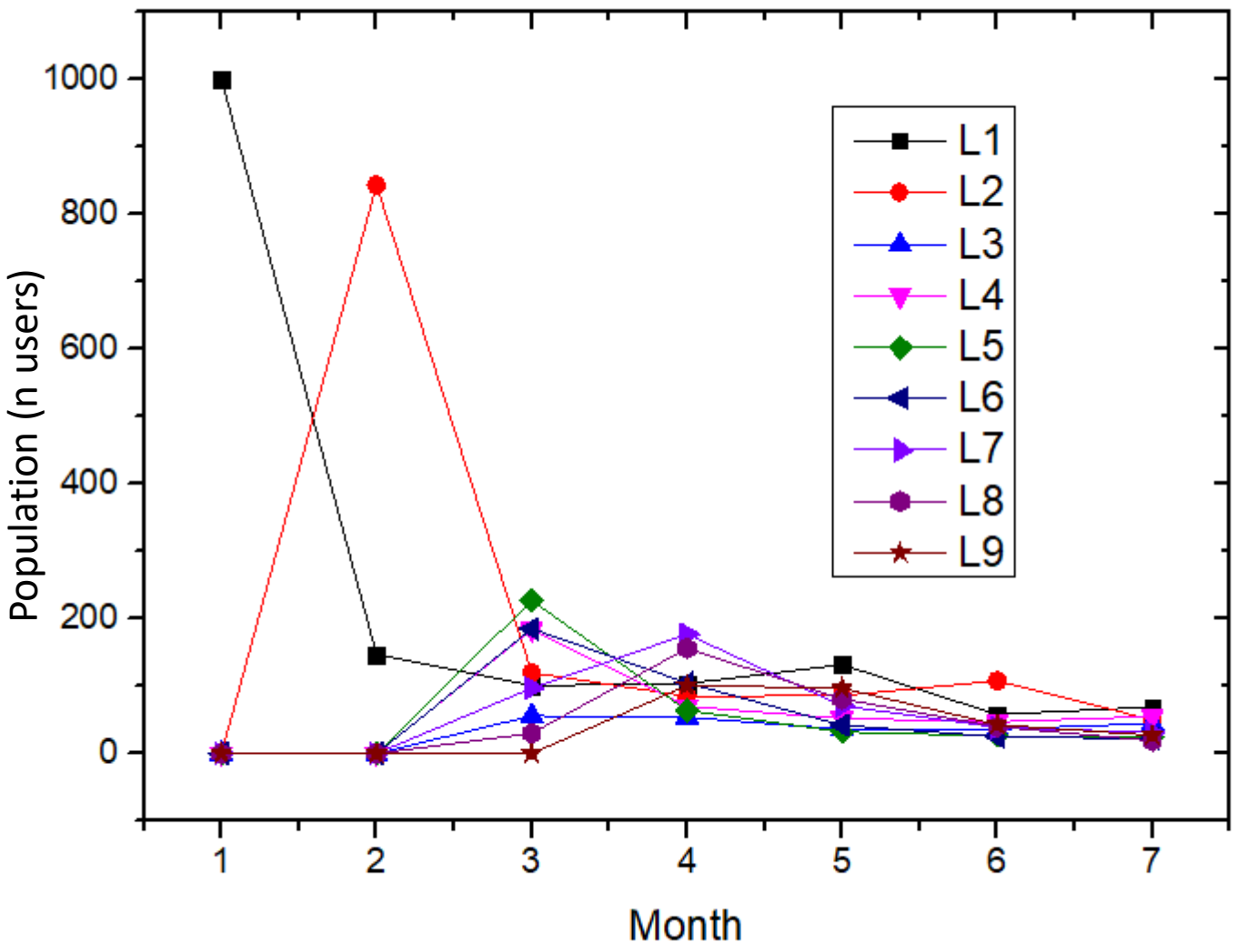


G(31.7,1)

The optimal prices vary for each gaussian (associated to lambda).

- *A priori* peak and dispersion (width) have been changed.
- *A posteriori* the fact that this position of the peak is the center of gravity of intermediate Gaussians has been verified.

As proceeding:
Useful Insights

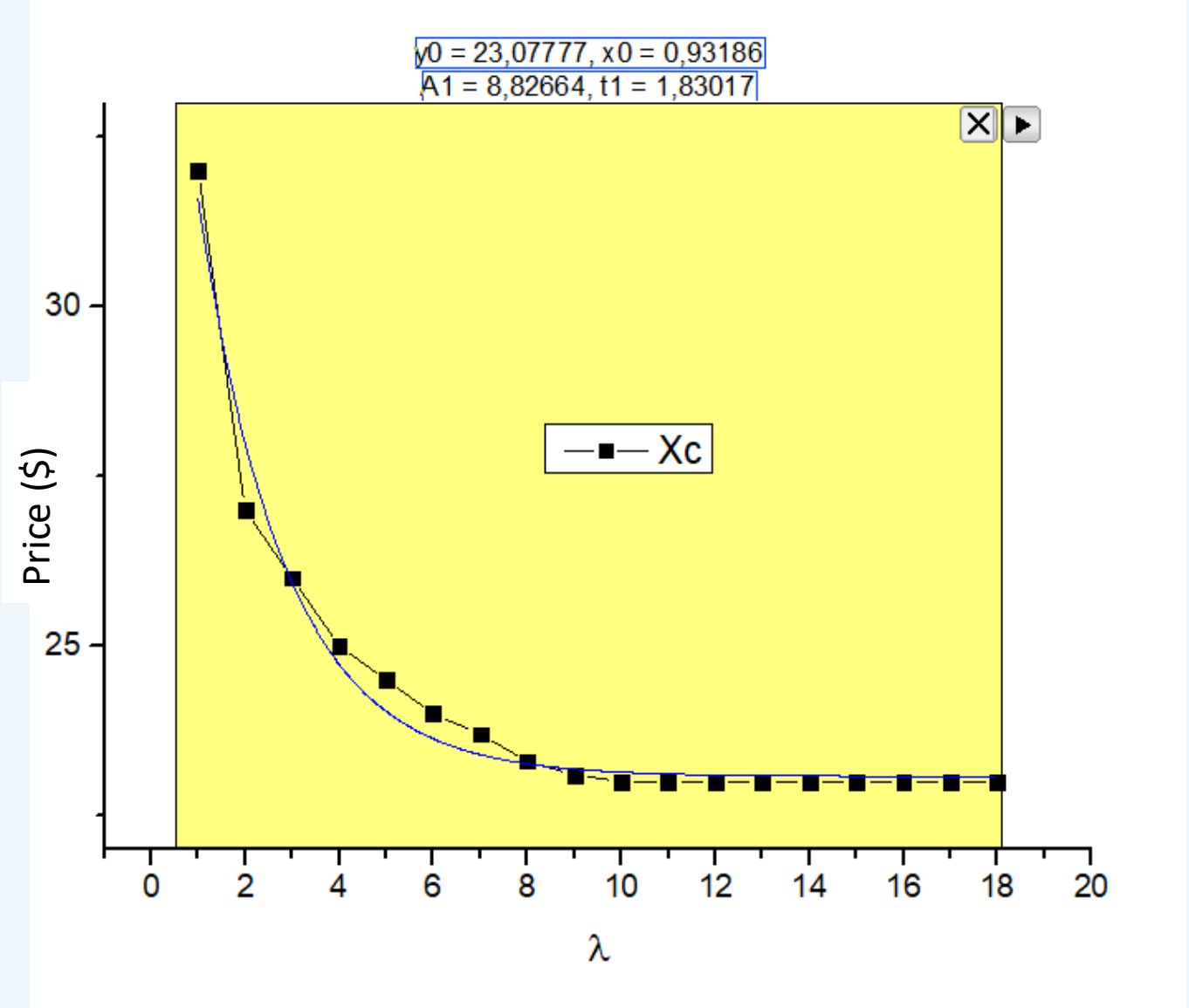


Population
Homogeneization

As the user population matures (lambda), the population begins to homogenize.

As proceeding:
Useful Insights

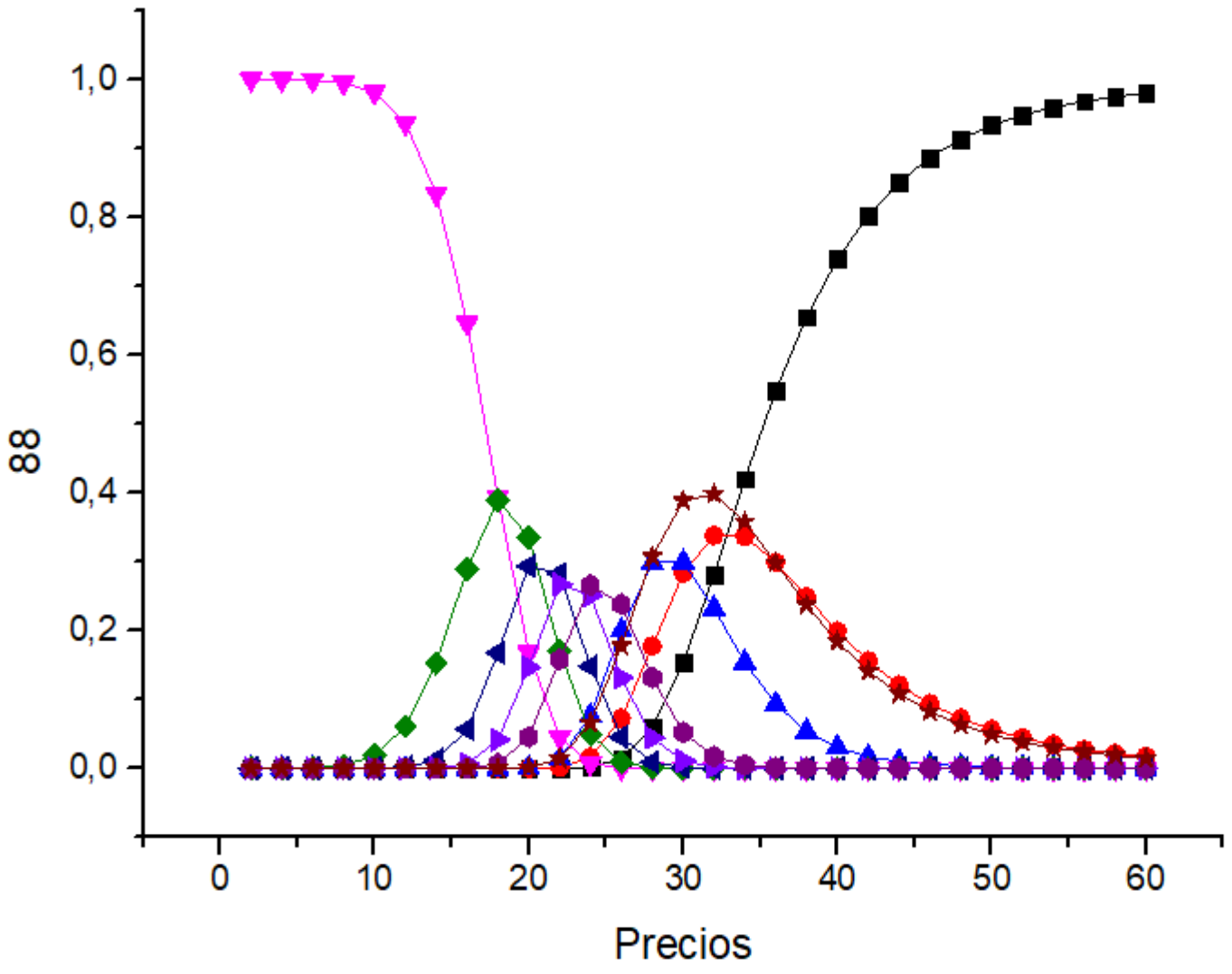
Price Stabilization

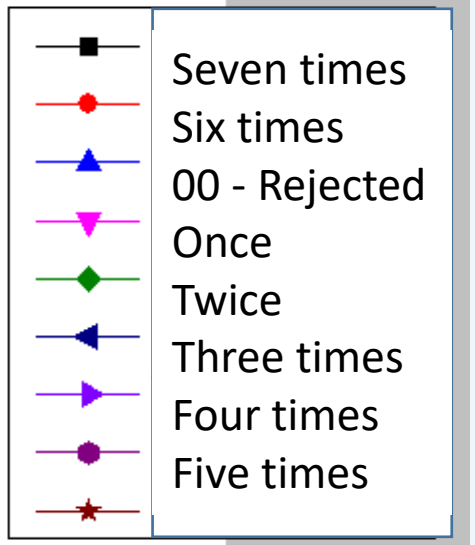


Optimal Gaussian peaks for lambda (0:18) fitted to exponential decay with offset 23.

As with the population distribution, prices stabilize too .

An example of user matureness process: lamda=8



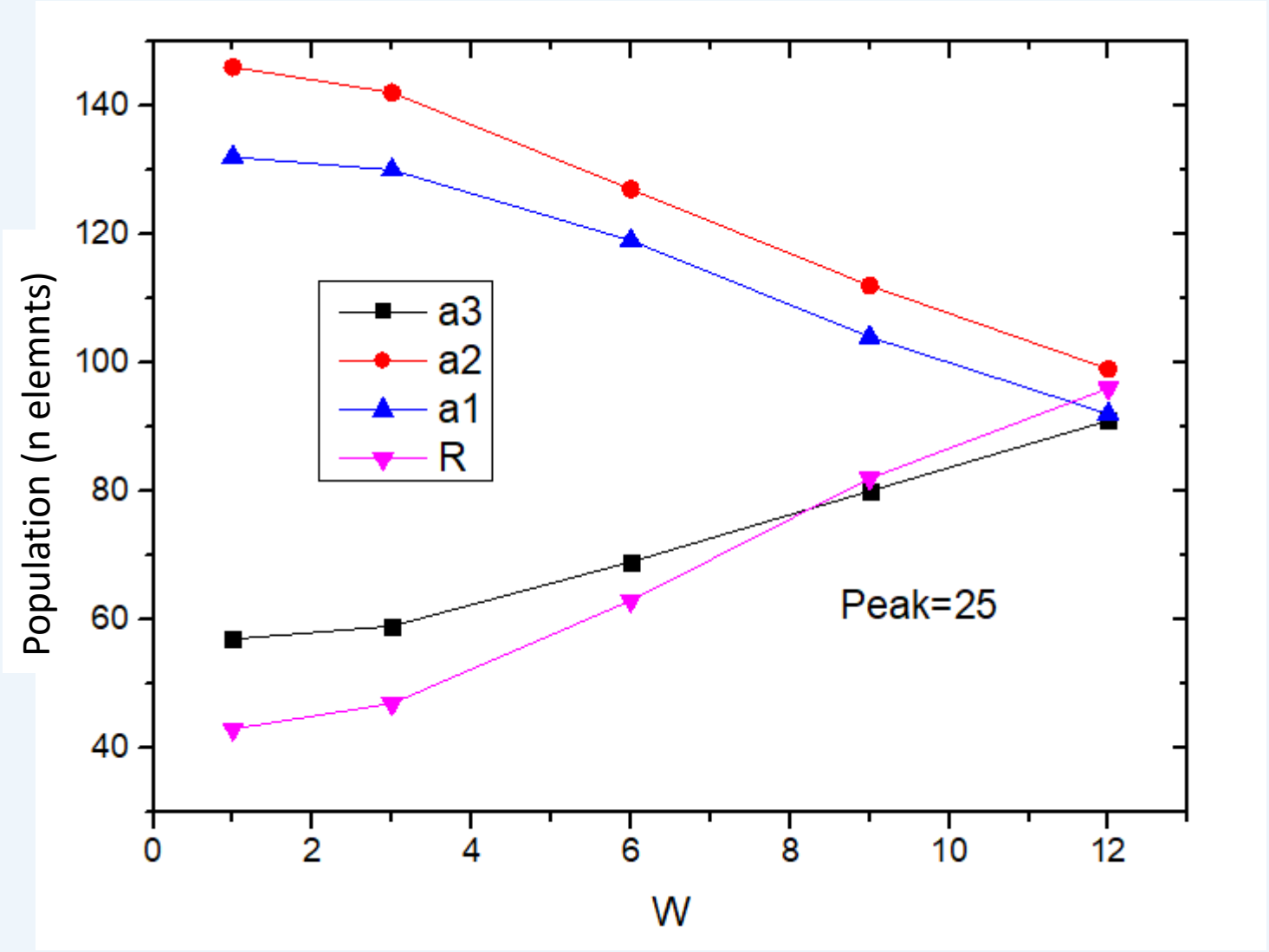


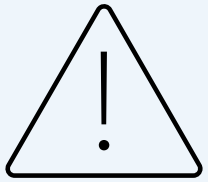
Example of lambda 8 subpopulation.

- Logistic each time shifted to the right.
- Probability of all cases being rejected grows smaller.

How is profit made?

Optimizing those app users prices from the intermediate Gaussian distributions

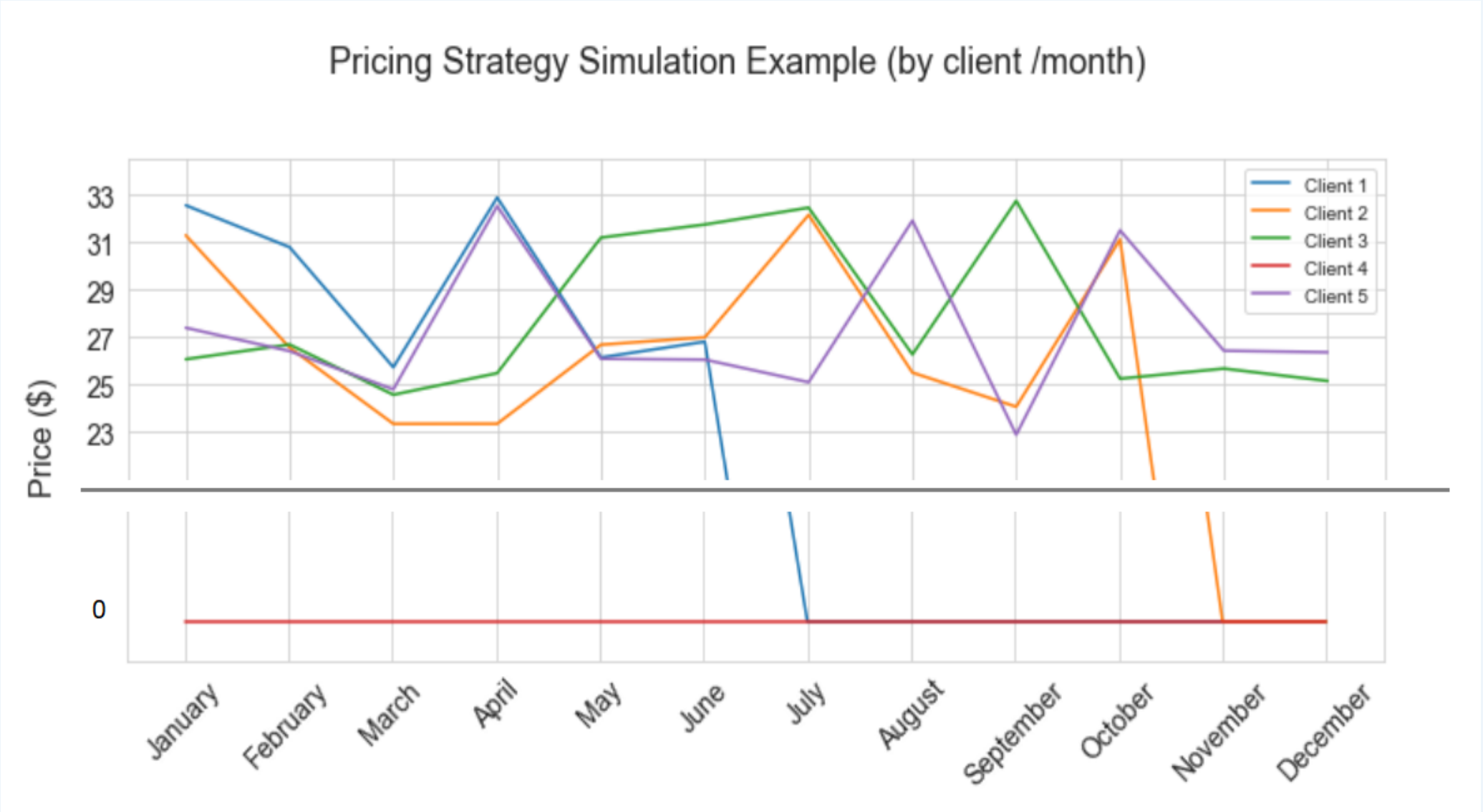




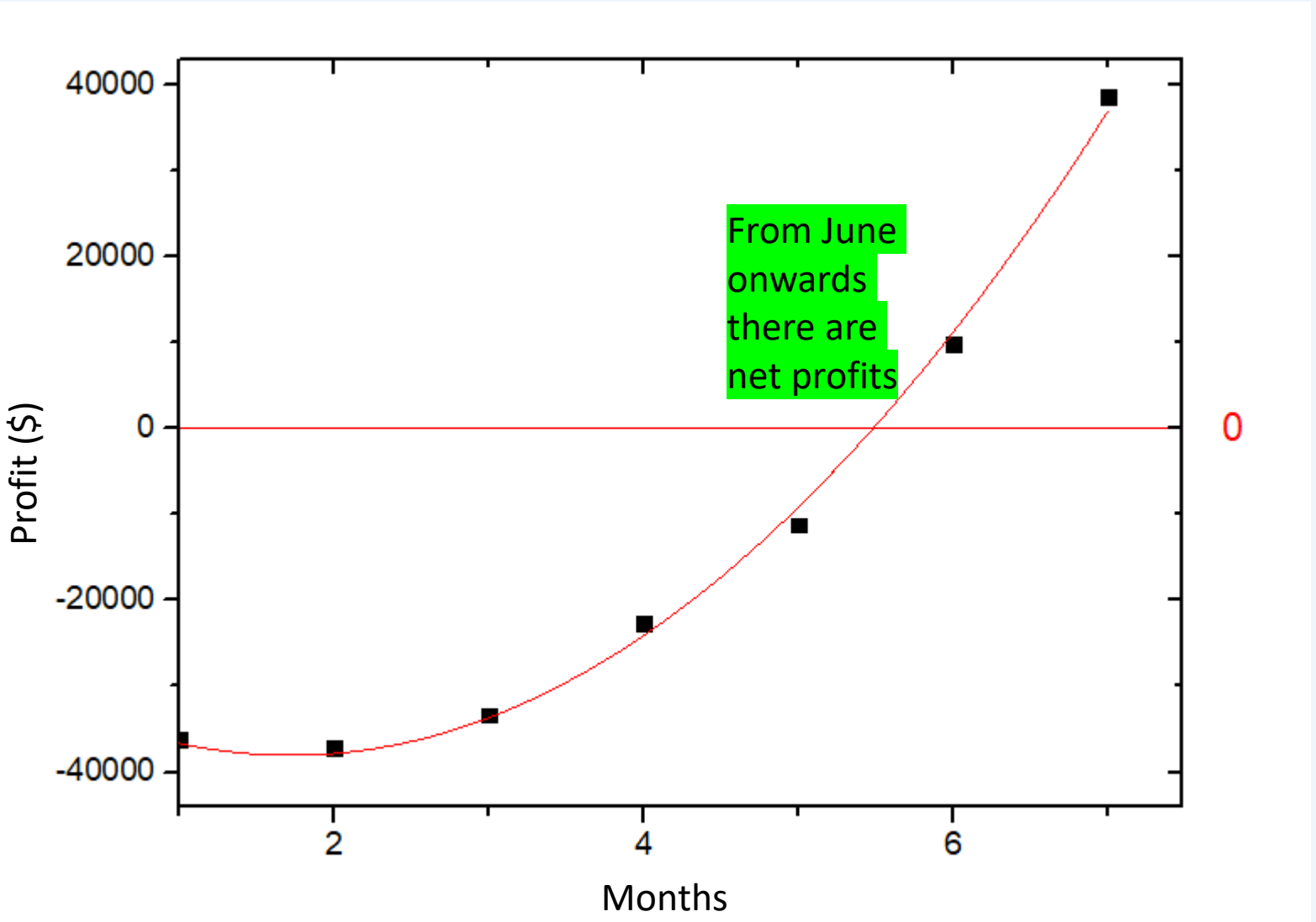
What if the dispersion is increased (w)?

- Logistics increase (non profitable distributions raise)
- Gaussians decrease (profitable distributions decline)

Pricing Strategy Appareance for individual app users:



Final Balance Forecast



Realized until July

- Until 0: Not profitable
- Parabolic growth
- Intercept: $-30,000 (new users) – January

		Value	Stan
B-FINAL	Intercept	-30000	
	B1	-9314,95098	
	B2	2698,47339	

Statistics

	B-FINAL
Number of Points	7
Degrees of Freedom	5
Residual Sum of Squares	1,16288E7
Adj. R-Square	0,9973

12-month earnings forecast:

246.876

Clipboard Health - Pricing Strategy (Code Phase)

Lamba processing proposal code

In [2]:

```python
# Proccess Explanation

# This code puts forward a solution for ClipHealth "Data Senior - Pricing" exercise.

# The goal of the code is to simulate the poisson distribution (lambda) according to the maturation process of
# the app users.

# The process is caged in 12 loops, each of them corresponding to a month in a year. (12 loops)
# The process is divided into five (5) stages printed in every loop:
# STAGE 1: The loop starts and every lambda condition is placed in an empty
#          array created to arrange every value representing a lambda condition.
# STAGE 2: According to the lambda condition, in every list as mentioned before, is being processed as
#          a poisson distribution where lamba = value of original launch_pool.
# STAGE 3: A montlhy DF has been created, and some outcomes are printed.
# STAGE 4: According to resources ("max resources pool 10,000" and "max reosurces per month = 1,000")
#          data is processed in two different ways: Adding more resources to launch pool
#                                                   Not adding any resources to launch pool
# STAGE 5: Extension of what has been mentioned before (stage 4)

# Need to add: Probability of dropping off from APP use according to driver acceptance/not acceptance.

# For troubleshooting, the outcomes are divided into 12 loops (MONTHS POW). Every loop has 5 stages that must be printed.


#_____________________________________________________________________________________________________

# Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
import datetime
from sklearn.utils import shuffle

#___________________________________________________________________________________________________

# Variables
# Adquisition pool

max_month = 1000
max_pool = 10000
launch_pool = np.random.poisson(1, 1000)

# DataFrame
price_month = pd.DataFrame()
price_month['January'] = []
price_month['February'] = []
price_month['March'] = []
price_month['April'] = []
price_month['May'] = []
price_month['June'] = []
price_month['July'] = []
price_month['August'] = []
price_month['September'] = []
price_month['October'] = []
price_month['November'] = []
price_month['December'] = []

# Months list

months = ("January","February","March","April","May","June","July",
        "August","September","October","November","December")



# Others

rider_pool = []
example_pricing_1 = []
example_pricing_2 = []
example_pricing_3 = []
example_pricing_4 = []
example_pricing_5 = []



#____________________________________________________________________________________________________

# Start the proccess

for month in months:



    # Declare variables where data will be storaged for each poisson condition in each loop

    cond0 = []
    cond1 = []
    cond2 = []
    cond3 = []
    cond4 = []
    cond5 = []
    cond6 = []
    cond7 = []
    cond8 = []
    cond9 = []
    cond9 = []
    cond10 = []
    cond11 = []
    cond12 = []
    cond13 = []
    cond14 = []
    cond15 = []
    cond16 = []
    cond17 = []
    cond18 = []
    cond19 = []
    cond20 = []
    cond21 = []
    cond22 = []
    cond23 = []
    cond24 = []
    cond25 = []
    cond26 = []
    cond27 = []
    cond28 = []
    cond29 = []
    cond30 = []
    cond31 = []
    cond32 = []


    # Appending for example



    # Grouped storage variables
    list_of_lists = (cond0, cond1, cond2, cond3, cond4, cond5, cond6, cond7, cond8, cond9, cond10
            ,cond11, cond12, cond13, cond14, cond15, cond16, cond17, cond18, cond19, cond20
            ,cond21, cond22, cond23, cond24, cond25, cond26, cond27, cond28, cond29, cond30
            ,cond31, cond32)

    # Header for after-messages
    print(" ---------- POW", month, "--------------------")


    # Iterate over every value that is included in lauch pool and append it to storage variables.


    for value in launch_pool:


        try:
            if value == 0:
                cond0.append(value)
            if value == 1:
                cond1.append(value)
            elif value == 2:
                cond2.append(value)
            elif value == 3:
                cond3.append(value)
            elif value == 4:
                cond4.append(value)
            elif value == 5:
                cond5.append(value)
            elif value == 6:
                cond6.append(value)
            elif value == 7:
                cond7.append(value)
            elif value == 8:
                cond8.append(value)
            elif value == 9:
                cond9.append(value)
            elif value == 10:
                cond10.append(value)
            elif value == 11:
                cond11.append(value)
            elif value == 12:
                cond12.append(value)
            elif value == 13:
                cond13.append(value)
            elif value == 14:
                cond14.append(value)
            elif value == 15:
                cond15.append(value)
            elif value == 16:
                cond16.append(value)
            elif value == 17:
                cond17.append(value)
            elif value == 18:
                cond18.append(value)
            elif value == 19:
                cond19.append(value)
            elif value == 20:
                cond20.append(value)
            elif value == 21:
                cond21.append(value)
            elif value == 22:
                cond22.append(value)
            elif value == 23:
                cond23.append(value)
            elif value == 24:
                cond24.append(value)
            elif value == 25:
                cond25.append(value)
            elif value == 26:
                cond26.append(value)
            elif value == 27:
                cond27.append(value)
            elif value == 28:
                cond28.append(value)
            elif value == 29:
                cond29.append(value)
            elif value == 30:
                cond30.append(value)
            elif value == 31:
                cond31.append(value)
            elif value == 32:
                cond32.append(value)

        except:
            pass

        # New variable that re-starts each time loop starts

        condition_union = []

    # Get data for DataFrame

    for element in list_of_lists:
        condition_union.append(len(element))

    # Drop Data Into DF so column in created

    price_month[month] = condition_union
    print("1. DATA ADDED TO DF. WORKS OK. N DATA ADDED:", price_month[month].sum())


    # Poisson Conditions are added. For every condition lamba = condition and
    # number elements = number of element storaged in storage variables

    try:
        print("2. POISSON REASIGNING WORKING OK")
        cond1 = list(np.random.poisson(cond1[0],len(cond1)))
    except:
        pass
    try:
        cond2 = list(np.random.poisson(cond2[0],len(cond2)))
    except:
        pass
    try:
        cond3 = list(np.random.poisson(cond3[0],len(cond3)))
    except:
        pass
    try:
        cond4 = list(np.random.poisson(cond4[0],len(cond4)))
    except:
        pass
    try:
        cond5 = list(np.random.poisson(cond5[0],len(cond5)))
    except:
        pass
    try:
        cond6 = list(np.random.poisson(cond6[0],len(cond6)))
    except:
        pass
    try:
        cond7 = list(np.random.poisson(cond7[0],len(cond7)))
    except:
        pass
    try:
        cond8 = list(np.random.poisson(cond8[0],len(cond8)))
    except:
        pass
    try:
        cond9 = list(np.random.poisson(cond9[0],len(cond9)))
    except:
        pass
    try:
        cond10 = list(np.random.poisson(cond10[0],len(cond10)))
    except:
        pass
    try:
        cond11 = list(np.random.poisson(cond11[0],len(cond11)))
    except:
        pass
    try:
        cond12 = list(np.random.poisson(cond12[0],len(cond12)))
    except:
        pass
    try:
        cond13 = list(np.random.poisson(cond13[0],len(cond13)))
    except:
        pass
    try:
        cond14 = list(np.random.poisson(cond14[0],len(cond14)))
    except:
        pass
    try:
        cond15 = list(np.random.poisson(cond15[0],len(cond15)))
    except:
        pass
    try:
        cond16 = list(np.random.poisson(cond16[0],len(cond16)))
    except:
        pass
    try:
        cond17 = list(np.random.poisson(cond17[0],len(cond17)))
    except:
        pass
    try:
        cond18 = list(np.random.poisson(cond18[0],len(cond18)))
    except:
        pass
    try:
        cond19 = list(np.random.poisson(cond19[0],len(cond19)))
    except:
        pass
    try:
        cond20 = list(np.random.poisson(cond20[0],len(cond20)))
    except:
        pass
    try:
        cond21 = list(np.random.poisson(cond21[0],len(cond21)))
    except:
        pass
    try:
        cond22 = list(np.random.poisson(cond22[0],len(cond22)))
    except:
        pass
    try:
        cond23 = list(np.random.poisson(cond23[0],len(cond23)))
    except:
        pass
    try:
        cond24 = list(np.random.poisson(cond24[0],len(cond24)))
    except:
        pass
    try:
        cond25 = list(np.random.poisson(cond25[0],len(cond25)))
    except:
        pass
    try:
        cond26 = list(np.random.poisson(cond26[0],len(cond26)))
    except:
        pass
    try:
        cond27 = list(np.random.poisson(cond27[0],len(cond27)))
    except:
        pass
    try:
        cond28 = list(np.random.poisson(cond28[0],len(cond28)))
    except:
        pass
    try:
        cond29 = list(np.random.poisson(cond29[0],len(cond29)))
    except:
        pass
    try:
        cond30 = list(np.random.poisson(cond30[0],len(cond30)))
    except:
        pass
    try:
        cond31 = list(np.random.poisson(cond31[0],len(cond31)))
    except:
        pass
    try:
        cond32 = list(np.random.poisson(cond32[0],len(cond32)))
    except:
        pass


    # Group again launch pool (transformed and proccessed) for next iteration

    launch_pool = cond1 + cond2 + cond3 + cond4 + cond5 + cond6 + cond7 + cond8 + cond9 + cond10 + cond11 + cond12 + cond13 + cond14 + cond15 + cond16 + cond17 + cond18 + cond19 +


    print("3. LEN LAUNCHPOOL", len(launch_pool))
    print("   ELIMINATED", len(cond0), "0s")

    example_pricing_1.append(shuffle(launch_pool)[0])
    example_pricing_2.append(shuffle(launch_pool)[1])
    example_pricing_3.append(shuffle(launch_pool)[2])
    example_pricing_4.append(shuffle(launch_pool)[3])
    example_pricing_5.append(shuffle(launch_pool)[4])

    print("   EXAMPLES HAS BEEN APPENDED")

    # Add new adquisition values untill 10000 pool is fullfiflled

    if max_pool > 0:

        new_adquisitions = 1000 - len(launch_pool)

        print("4. CURRENTLY MAX POOL > 0 CORRECTO")
        print("   STILL N", new_adquisitions, "TO 1000 ELEMENTS")


        launch_pool = np.append(launch_pool, np.random.poisson(1, new_adquisitions))
        max_pool = max_pool - new_adquisitions

        print("5. ADQUISITIONS ARE NOW HIGHER THAN 0")
        print("   new_adquisitons SIZE", new_adquisitions)
        print("   max pool SIZE", max_pool)


    else:
        print("5. MAX POOL HAS NO MORE ELEMENTS")
        pass


    ##############
    print( "\n \n")


####
#________________________________________________________________________________________________
price_month.to_excel("poisson_users.xlsx")
print( "\n \n Saved as poisson_users.xlsx")

price_month
```

```
 ---------- POW January --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 621
   ELIMINATED 379 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 379 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 379
   max pool SIZE 9621



 ---------- POW February --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 704
   ELIMINATED 296 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 296 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 296
   max pool SIZE 9325



 ---------- POW March --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 746
   ELIMINATED 254 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 254 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 254
   max pool SIZE 9071



 ---------- POW April --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 732
   ELIMINATED 268 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 268 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 268
   max pool SIZE 8803



 ---------- POW May --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 762
   ELIMINATED 238 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 238 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 238
   max pool SIZE 8565



 ---------- POW June --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 791
   ELIMINATED 209 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 209 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 209
   max pool SIZE 8356



 ---------- POW July --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 800
   ELIMINATED 200 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 200 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 200
   max pool SIZE 8156



 ---------- POW August --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 775
   ELIMINATED 225 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 225 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 225
   max pool SIZE 7931



 ---------- POW September --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 807
   ELIMINATED 193 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 193 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 193
   max pool SIZE 7738



 ---------- POW October --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 819
   ELIMINATED 181 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 181 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 181
   max pool SIZE 7557



 ---------- POW November --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 813
   ELIMINATED 187 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 187 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 187
   max pool SIZE 7370



 ---------- POW December --------------------
1. DATA ADDED TO DF. WORKS OK. N DATA ADDED: 1000
2. POISSON REASIGNING WORKING OK
3. LEN LAUNCHPOOL 800
   ELIMINATED 200 0s
   EXAMPLES HAS BEEN APPENDED
4. CURRENTLY MAX POOL > 0 CORRECTO
   STILL N 200 TO 1000 ELEMENTS
5. ADQUISITIONS ARE NOW HIGHER THAN 0
   new_adquisitons SIZE 200
   max pool SIZE 7170




 Saved as poisson_users.xlsx
```

Out[2]:

	January	February	March	April	May	June	July	August	September	October	November	December
0	379	296	254	268	238	209	200	225	193	181	187	200
1	374	344	343	304	278	257	262	228	240	214	227	223
2	172	190	182	188	192	217	177	150	162	178	163	150
3	61	100	116	91	124	111	114	128	127	122	102	119
4	12	32	30	49	66	57	79	83	76	80	80	66
5	2	23	39	36	27	58	50	53	55	57	53	50
6	0	7	17	25	25	34	30	43	39	53	44	41
7	0	5	10	11	13	18	26	26	25	20	36	30
8	0	1	4	12	11	11	15	13	21	24	21	21
9	0	2	3	3	10	6	14	10	9	12	20	21
10	0	0	1	2	6	7	6	6	8	12	7	13
11	0	0	0	2	2	5	10	7	7	12	13	23
12	0	0	1	5	3	1	3	8	6	9	12	3
13	0	0	0	2	1	3	4	4	9	2	10	5
14	0	0	0	0	2	2	2	4	6	4	5	4
15	0	0	0	1	0	2	1	3	8	3	4	6
16	0	0	0	0	1	1	1	1	1	2	3	5
17	0	0	0	0	0	0	2	2	4	5	0	6
18	0	0	0	1	0	1	2	2	1	3	3	2
19	0	0	0	0	1	0	1	0	1	2	3	3
20	0	0	0	0	0	0	0	1	0	3	0	2
21	0	0	0	0	0	0	1	1	1	1	2	1
22	0	0	0	0	0	0	0	1	0	0	2	0
23	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	1	0	0	2
25	0	0	0	0	0	0	0	0	0	0	1	2
26	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	1	0	1
29	0	0	0	0	0	0	0	1	0	0	0	1
30	0	0	0	0	0	0	0	0	0	0	1	0
31	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	1	0