Data1201 Final project

December 8, 2023

Colin Watson
Data 1201 Final Project
12/8/2023
Exploring data with Bloons

1 Introduction

Bloons Tower Defense 6 is a tower defense game created by Ninja Kiwi in 2018. In this game, bloons move through preset tracks, and the player must place various towers to pop the bloons before they reach the end of the track. Each bloon you pop gives you one dollar to spend on upgrading your monkeys, and the player must carefully manage their money to buy a defense strong enough to defeat all the waves of bloons. This portfolio is up to date as of Update 39, but some parts of it will inevitably become outdated, as BTD6 is still getting frequent updates, of which often change prices of upgrades. And in case you are wondering, yes, balloon is misspelled in the game and all of its media.

2 The Question

Each upgrade costs a different amount, and they can vary greatly in power. For example, the upgrade Sharp Shots for the Dart Monkey costs $140, and the upgrade Laser Blasts for the Super Monkey costs $2,500. Despite the vastly different prices for these upgrades, they both do the same thing: add 1 pierce to the main projectile. Each tower has different upgrades not availible to the other towers, which make it difficult to make accurate comparisons between towers.

One thing that could help in making these comparisons is finding the average cost of an upgrade, which could be used as a baseline. My goal is to find what the average price of an upgrade in BTD6 is, so that can be used as a baseling for further comparisons.

When brainstorming ideas for this presentation, I recalled watching a YouTube video about finding the statistically most average weapon in Terraria (Link: https://www.youtube.com/watch?v=dC7UL0GIuI4). This inspired me to do a similar thing for Bloons. Eventually I decided on finding the average upgrade price.

3 Getting the Data

In BTD6, each tower has three upgrade paths; the top path, middle path, and bottom path. The player can choose one of these paths to upgrade up to five times, then one of the other two paths up to two times, paying for each upgrade as it is purchased. Some towers also have a Paragon, which is like a sixth tier, specifically designed to be strong against boss bloons. However, not all towers have a paragon yet (they are a relatively new addition to the game) and they are very different from standard upgrades. Not only are they vastly more expensive than other towers, they get more powerful based on how many towers of that type you had before the upgrade, and how much damage those towers did. Paragons are very complicated, and therefore way outside the scope of this project.

The data for this project was sourced from the Bloons TD6 Fandom Wikipedia, which got all its information directly from the game (Link: https://bloons.fandom.com/wiki/Upgrades). I then used Microsoft Excel's web scraping tool to easily get the data into a table. The upgrade names and prices were combined into the same cell when exported, so some time was needed to separate the values and get it ready for processing.

Because all data used in this project is from publicly avaliable soources and pertains to a video game, not real life, there are not too many ethical considerations to worry about. However, if this data were related to real life situations, the analysis performed here would need to be much deeper and explore more of the connections between towers to make better comparisons.

[1]:
```
# Initialize all the libraries needed
import numpy as np
from datascience import *

import matplotlib.pyplot as plt
plt.style.use("ggplot")
%matplotlib inline
```

[2]:
```
# Load the data into a table
bloons_upgrades_full = Table.read_table('Final_project_data/BloonsUpgrades.csv')
bloons_upgrades_full
```

[2]:
```
Tower             | BaseCost | Tier1Name        | Tier1Cost | Tier2Name
| Tier2Cost | Tier3Name         | Tier3Cost | Tier4Name              | Tier4Cost |
Tier5Name              | Tier5Cost | ParagonName         | ParagonCost
nan               | 0        | Sharp Shots      | 140       | Razor Sharp Shots
| 220       | Spike-O-Pult      | 300       | Juggernaut             | 1800      |
Ultra-Juggernaut       | 15000     | nan                 | 0
Dart Monkey       | 200      | Quick Shots      | 100       | Very Quick Shots
| 190       | Triple Shot       | 400       | Super Monkey Fan Club  | 8000      |
Plasma Monkey Fan Club | 45000     | Apex Plasma Master  | 150000
nan               | 0        | Long Range Darts | 90        | Enhanced Eyesight
| 200       | Crossbow          | 625       | Sharp Shooter          | 2000      |
Crossbow Master        | 21500     | nan                 | 0
nan               | 0        | Improved Rangs   | 200       | Glaives
| 280       | Glaive Ricochet   | 1200      | M.O.A.R Glaives        | 3000      |
Glaive Lord            | 29400     | nan                 | 0
Boomerang Monkey  | 325      | Faster Throwing  | 175       | Faster Rangs
| 250       | Bionic Boomerang  | 1450      | Turbo Charge           | 4200      |
Perma Charge           | 35000     | Glaive Dominus      | 275000
```

```
nan              | 0          | Long Range Rangs | 100        | Red Hot Rangs
| 300        | Kylie Boomerang  | 1300       | MOAB Press             | 2400       |
MOAB Domination        | 50000      | nan                  | 0
nan              | 0          | Bigger Bombs     | 350        | Heavy Bombs
| 650        | Really Big Bombs | 1200       | Bloon Impact           | 3600       |
Bloon Crush            | 55000      | nan                  | 0
Bomb Shooter     | 525        | Faster Reload    | 250        | Missile Launcher
| 400        | MOAB Mauler      | 1100       | MOAB Assassin          | 3200       |
MOAB Eliminator        | 25000      | nan                  | 0
nan              | 0          | Extra Range      | 200        | Frag Bombs
| 300        | Cluster Bombs    | 800        | Recursive Cluster      | 2800       |
Bomb Blitz             | 35000      | nan                  | 0
nan              | 0          | Faster Shooting  | 150        | Even Faster
Shooting | 300        | Hot Shots        | 600        | Ring of Fire           |
3500       | Inferno Ring           | 45500      | nan                  | 0
… (59 rows omitted)
```

Each tier upgrade has a column for its upgrade name and the price, as well as one for the base tower and cost. The rows above and below the towers name are null and costs are 0 as there are more upgrade paths than towers, leading to some of the rows either needing to repeat or be empty. I set it up this way, as those empty values can easily be filtered out. There is also a column for the towers which have a paragon, which is just there for future use.

[3]:
```
# Select only the columns we need for costs
bloons_upgrades_costs = bloons_upgrades_full.select('BaseCost','Tier1Cost',
                                                                          ␣
 ↪'Tier2Cost','Tier3Cost','Tier4Cost','Tier5Cost')
bloons_upgrades_costs
```

[3]:
```
BaseCost | Tier1Cost | Tier2Cost | Tier3Cost | Tier4Cost | Tier5Cost
0        | 140       | 220       | 300       | 1800      | 15000
200      | 100       | 190       | 400       | 8000      | 45000
0        | 90        | 200       | 625       | 2000      | 21500
0        | 200       | 280       | 1200      | 3000      | 29400
325      | 175       | 250       | 1450      | 4200      | 35000
0        | 100       | 300       | 1300      | 2400      | 50000
0        | 350       | 650       | 1200      | 3600      | 55000
525      | 250       | 400       | 1100      | 3200      | 25000
0        | 200       | 300       | 800       | 2800      | 35000
0        | 150       | 300       | 600       | 3500      | 45500
… (59 rows omitted)
```

[4]:
```
# Getting a table with only base costs, and filtering out the empty values
bloons_base_costs = bloons_upgrades_costs.select('BaseCost').
 ↪where('BaseCost',are.not_equal_to(0))
bloons_base_costs
```

[4]:
```
BaseCost
200
325
525
280
500
225
350
325
500
800
… (13 rows omitted)
```

[5]:
```python
# Getting a table with only tier 1 costs
bloons_tier1_costs = bloons_upgrades_costs.select('Tier1Cost')
bloons_tier1_costs
```

[5]:
```
Tier1Cost
140
100
90
200
175
100
350
250
200
150
… (59 rows omitted)
```

[6]:
```python
# Getting a table with only tier 2 costs
bloons_tier2_costs = bloons_upgrades_costs.select('Tier2Cost')
bloons_tier2_costs
```

[6]:
```
Tier2Cost
220
190
200
280
250
300
650
400
300
300
… (59 rows omitted)
```

[7]:
```python
# Getting a table with only tier 3 costs
bloons_tier3_costs = bloons_upgrades_costs.select('Tier3Cost')
bloons_tier3_costs
```

[7]:
```
Tier3Cost
300
400
625
1200
1450
1300
1200
1100
800
600
… (59 rows omitted)
```

[8]:
```python
# Getting a table with only tier 4 costs
bloons_tier4_costs = bloons_upgrades_costs.select('Tier4Cost')
bloons_tier4_costs
```

[8]:
```
Tier4Cost
1800
8000
2000
3000
4200
2400
3600
3200
2800
3500
… (59 rows omitted)
```

[9]:
```python
# Getting a table with only tier 5 costs
bloons_tier5_costs = bloons_upgrades_costs.select('Tier5Cost')
bloons_tier5_costs
```

[9]:
```
Tier5Cost
15000
45000
21500
29400
35000
50000
55000
25000
```

```
35000
45500
… (59 rows omitted)
```

4 Exploring the Data

First we need sums of the price for each upgrade in a tier.

[10]:
```
# Getting the sum of base costs
base_cost_total = np.sum(bloons_base_costs.column('BaseCost'))
base_cost_total
```

[10]:
```
15655
```

[11]:
```
# Getting the sum of tier 1 costs
tier1_cost_total = np.sum(bloons_tier1_costs.column('Tier1Cost'))
tier1_cost_total
```

[11]:
```
25975
```

[12]:
```
# Getting the sum of tier 2 costs
tier2_cost_total = np.sum(bloons_tier2_costs.column('Tier2Cost'))
tier2_cost_total
```

[12]:
```
41945
```

[13]:
```
# Getting the sum of tier 3 costs
tier3_cost_total = np.sum(bloons_tier3_costs.column('Tier3Cost'))
tier3_cost_total
```

[13]:
```
166700
```

[14]:
```
# Getting the sum of tier 4 costs
tier4_cost_total = np.sum(bloons_tier4_costs.column('Tier4Cost'))
tier4_cost_total
```

[14]:
```
612955
```

[15]:
```
# Getting the sum of tier 5 costs
tier5_cost_total = np.sum(bloons_tier5_costs.column('Tier5Cost'))
tier5_cost_total
```

[15]:
```
3419600
```

Now that we have the sum for each tiers total cost, we can find the average.

```
[16]: # Getting the total upgrade cost
      total_upgrade_cost = base_cost_total + tier1_cost_total + tier2_cost_total +␣
       ↪tier3_cost_total + tier4_cost_total + tier5_cost_total
      total_upgrade_cost
```

```
[16]: 4282830
```

```
[17]: # Getting the cost to have one of each monkey on the field at the same time.
      # Since each tier is a prerequesite for the one after it, we may need to add␣
       ↪the cost for the tower multiple times.
      # The base cost needs to be added 12 times total, since each tower would need␣
       ↪to be placed thrice.
      all_monkeys_cost = (tier5_cost_total + tier4_cost_total * 2 + tier3_cost_total␣
       ↪* 3 + tier2_cost_total * 4 +
                          tier1_cost_total * 5 + base_cost_total *  3 * 6)
      all_monkeys_cost
```

```
[17]: 5725055
```

```
[18]: # Dividing the total upgrade cost by 15 upgrades for 23 towers, plus the 23␣
       ↪towers at base.
      # The game always rounds prices down to a number that ends in 5 or 0, so the␣
       ↪price would go to the lower value printed.
      average_upgrade_cost = total_upgrade_cost / (15 * 23 + 23)
      print(average_upgrade_cost)
      average_upgrade_cost = int(average_upgrade_cost) - 3
      average_upgrade_cost
```

```
11638.125
```

```
[18]: 11635
```

$11,635 is our number for the average cost of an upgrade, and the answer to our question.

We can also find the averages of each tier individually.

```
[19]: np.mean(bloons_base_costs.column(0))
```

```
[19]: 680.6521739130435
```

```
[20]: np.mean(bloons_tier1_costs.column(0))
```

```
[20]: 376.44927536231882
```

```
[21]: np.mean(bloons_tier2_costs.column(0))
```

```
[21]: 607.89855072463763
```

[22]:
```
np.mean(bloons_tier3_costs.column(0))
```

[22]:
```
2415.942028985507
```

[23]:
```
np.mean(bloons_tier4_costs.column(0))
```

[23]:
```
8883.4057971014499
```

[24]:
```
np.mean(bloons_tier5_costs.column(0))
```

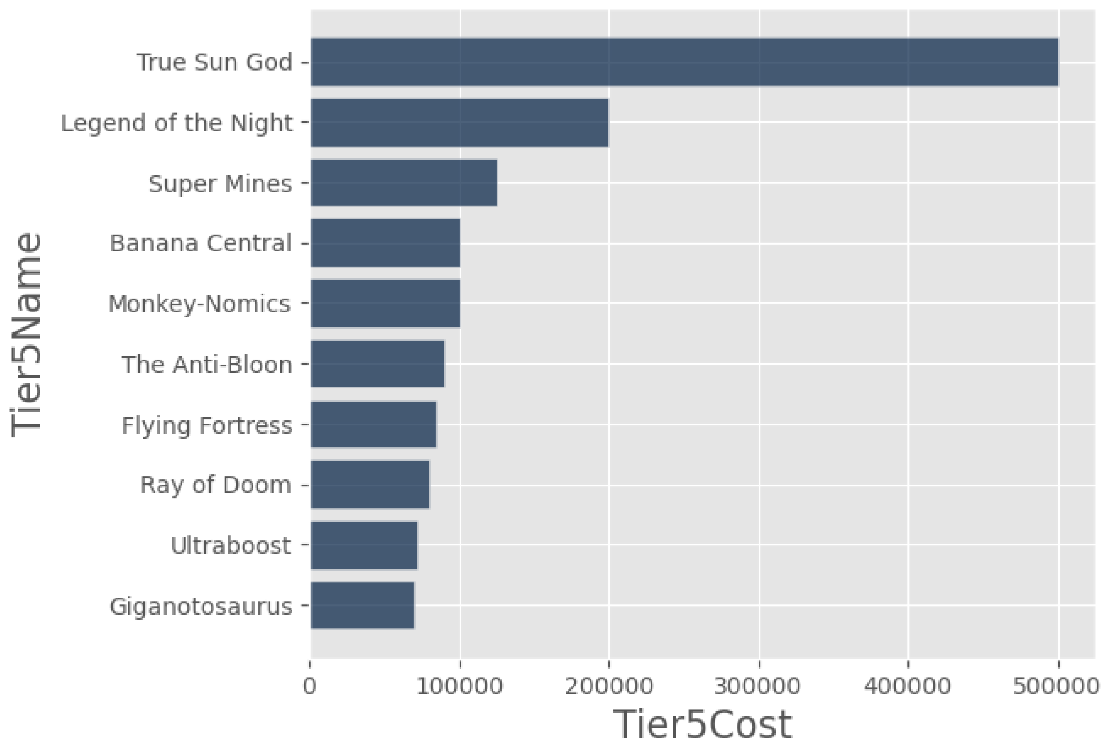
[24]:
```
49559.420289855072
```

Here is a graph of the 10 most expensive tier 5 upgrades.

[25]:
```
tier5_10biggest_upgrades = bloons_upgrades_full.select('Tier5Name','Tier5Cost').
  ↪sort('Tier5Cost', descending=True).take(np.arange(10))
tier5_10biggest_upgrades
```

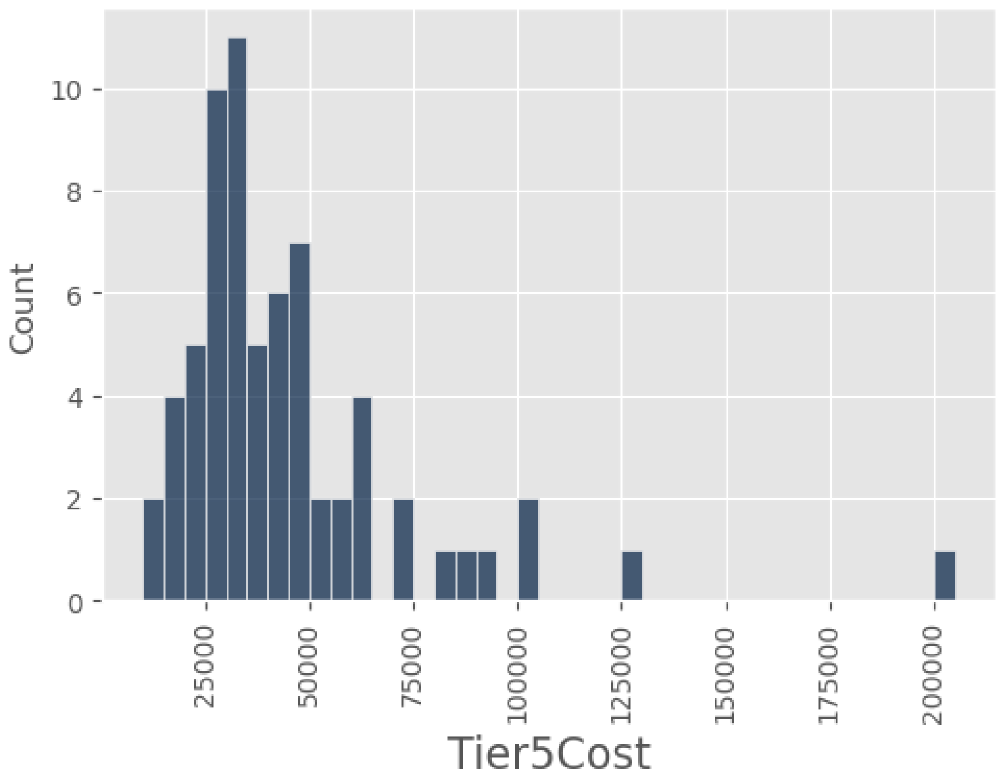
[25]:
```
Tier5Name            | Tier5Cost
True Sun God         | 500000
Legend of the Night  | 200000
Super Mines          | 125000
Banana Central       | 100000
Monkey-Nomics        | 100000
The Anti-Bloon       | 90000
Flying Fortress      | 85000
Ray of Doom          | 80000
Ultraboost           | 72000
Giganotosaurus       | 70000
```

[26]:
```
tier5_10biggest_upgrades.barh('Tier5Name')
```

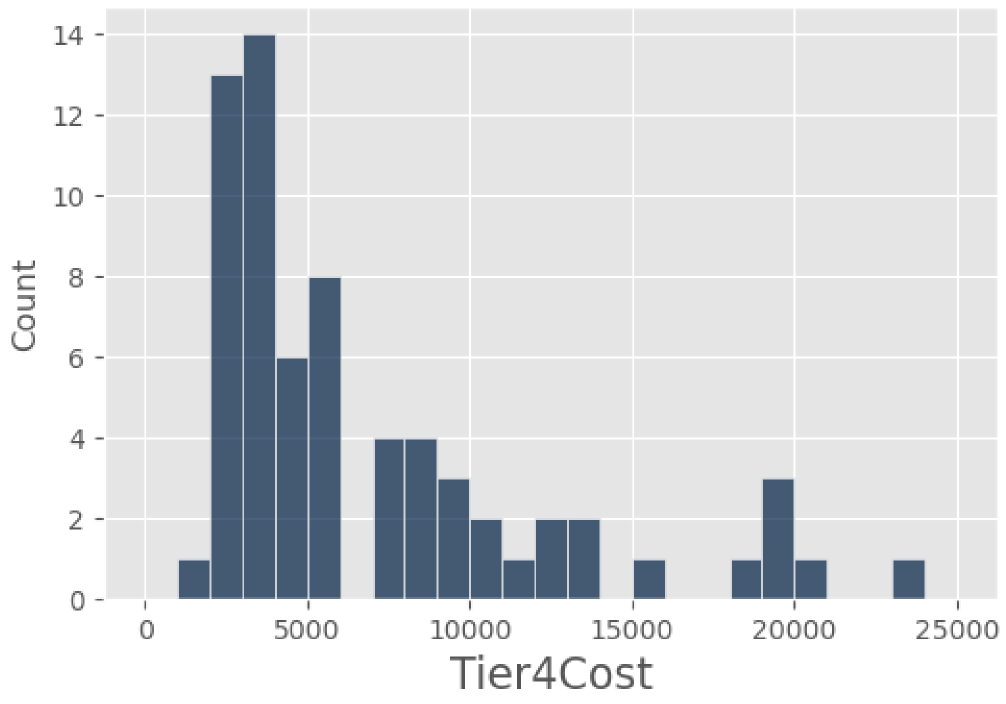


We can also make histograms showing the price ranges most upgrades fall in. In each graph besides tier 1 and 2, the most expensive towers is cut off in order to make the graph more readable.

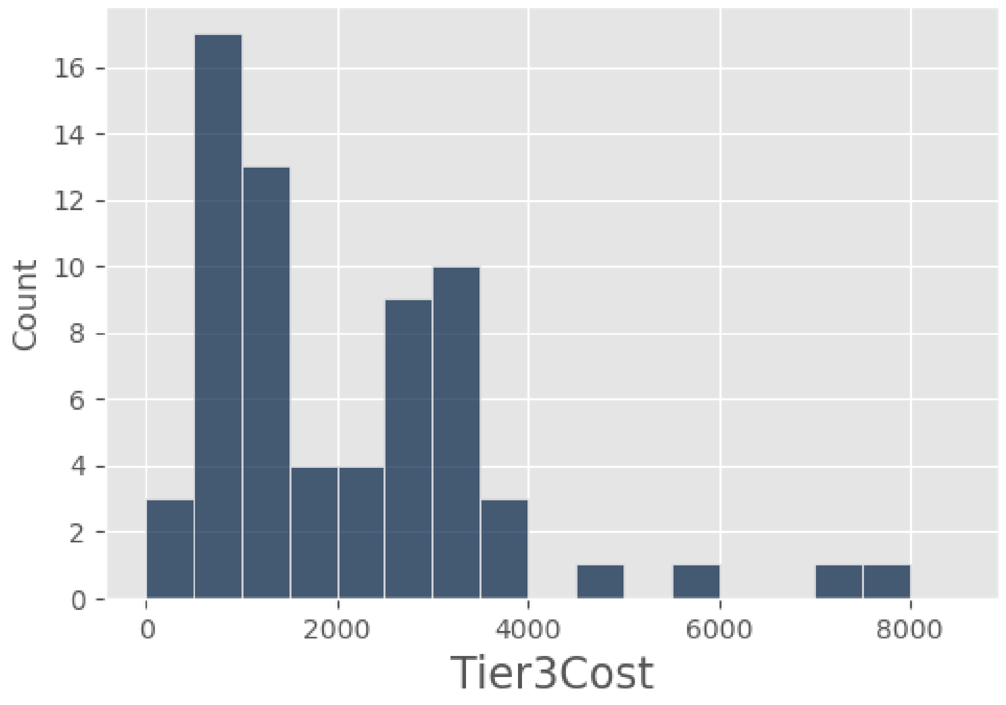
```
[31]: bloons_tier5_costs.hist('Tier5Cost', density = False, bins = np.arange(10000,␣
      ↪210000, 5000))
```



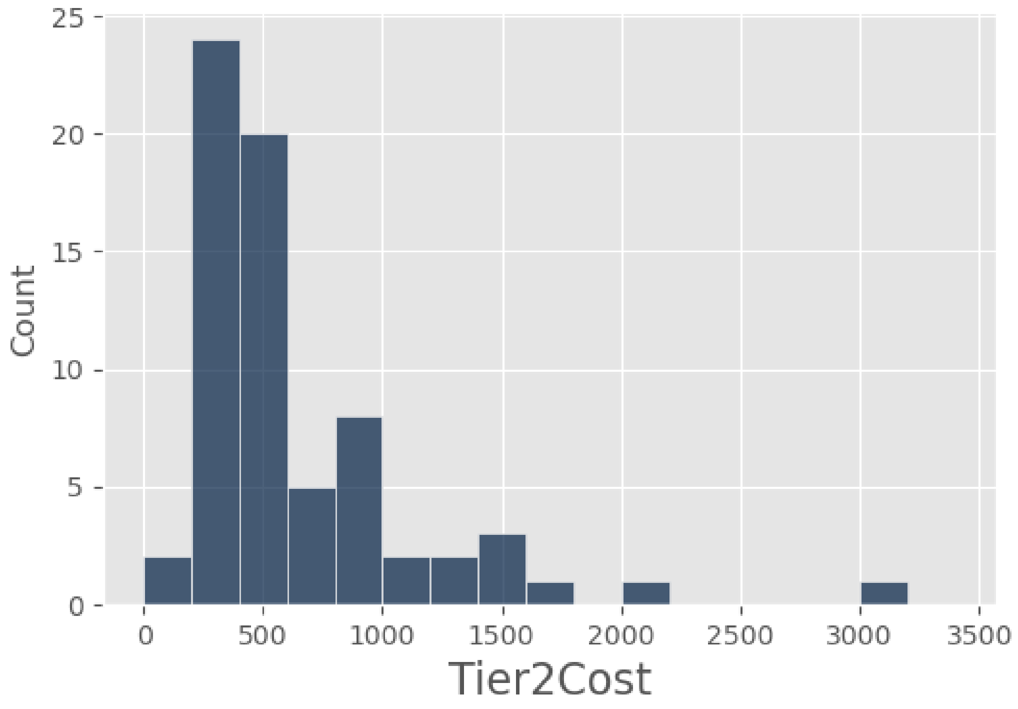
```
[28]: bloons_tier4_costs.hist('Tier4Cost', density = False, bins = np.arange(0,␣
      ↪26000, 1000))
```



[29]:
```
bloons_tier3_costs.hist('Tier3Cost', density = False, bins = np.arange(0, 9000,␣
↪500))
```



[30]:
```
bloons_tier2_costs.hist('Tier2Cost', density = False, bins = np.arange(0, 3500,␣
↪200))
```



5 Conclusion

In my opinion as a fairly experienced player of BTD6, the values I ended up with make a lot of sense. The costs of the top path Super Monkey upgrades being very high for each tier certainly increase the average values, but I would expect to pay around $12,000 for an average tower that does well in the midgame. While these numbers don't work well forcomparing towers directly yet, they can certainly make furture comparisons easier, knowing whether any particular upgrade is expensive or not.