

Roller Coaster Frenzy

Summary

In tackling the presented problem, we first defined the ranking of a roller coaster based on how thrilling it is, where thrill is measured by adrenaline rush. This logic was used to determine which objective specifications of a roller coaster contributed to how much thrill a roller coaster invokes, where we narrowed down to the following five parameters: drop height, speed, G-force, number of inversions, and the duration. In the given list of roller coasters and their parameters, a sizeable portion of the data was not available. To account for this, multiple regression models were created to relate the parameters with each other in order to fill out the missing data. After that, the team proceeded to develop a ranking model where all of the data is unweighted, but still scaled (equal weighting). In doing so, the team created a series of three models, each of which produced a base score for the amount of thrill of each roller coaster. These base scores were then used to rank the roller coasters.

Our first model used the average of each parameter to scale the data of each category, and then added them up to get a base score. The second model followed the same reasoning as Model 1, except it used the median of each parameter to scale the data. The third model utilized functions for each parameter to get an index value / score from zero to one by plugging in each roller coaster data value into each respective function. The functions were derived using the maximum values of each parameter, which were determined by the given specifications.

The resulting rankings were then compared with data from a series of online external sources and ranking systems to determine the objective model's accuracy with subjective opinions on the roller coasters. We found that the rankings generated by our algorithms were somewhat similar to the rankings of multiple online sources, which shows that our algorithm closely reflects the opinions of the public, despite being an objective model.

We proceeded on to design an outline for a mobile application which determines the best roller coasters for the user based on his or her preferences. The app utilizes Model 2 to create a list of the best roller coaster for the user, except it incorporates the user's preferences as multipliers to the base scores. This weighs the parameters and creates a ranking of roller coasters customized to each user.

News Release

We, the mathematicians of team 8600, have modeled a solution to the age old question: what are the top 10 roller coasters in the world? We set out to answer this question in the most objective way possible.

One of the problems that we ran into in the beginning was the absence of data on roller coasters. We tried as best as we could to fill in the missing data points with predictions that we based on trends from roller coasters with complete data.

After filling in the data as much as we could, we tried three ways to determine the top 10 roller coasters. Eventually, we settled on one solution. In this solution, we gave a score to each roller coaster for each category by comparing its value to the median value of that category. We then added up the scores of each category for each roller coaster and ranked the roller coasters starting with the coasters with the highest scores.

The results we came up with were, from first to last: Smiler, Steel Dragon 2000, Kingda Ka, Medusa, Leviathan, Top Thrill Dragster, Coaster Through the Clouds, Banshee, Scream!, and Alpengeist.

These are the results of our unweighted model. If you have certain things you like most about roller coasters, our model may also be adapted to your preferences. We have also designed an interactive app to help find the best rollercoaster for anybody who uses it. Our app uses the algorithm we designed to determine the top 10 roller coasters to find the roller coaster that best fits the preferences of the user.

The app asks a series of questions to get to know the preferences of the user and weights the different data sections accordingly. The app then shows you which roller coasters in your area best fit your preferences. For the more adventurous roller coaster rider, there is a search bar and advanced search available which allows the user to more precisely choose which roller coaster to ride.

Table of Contents

Problem Statement + Introduction.....	
Assumptions.....	
How Ranking is Determined.....	
Parameter Relevance to Algorithm+App.....	
Filling in Missing Data.....	
List of Assumptions:.....	
Model 1 - Algorithm (Means of Criteria).....	
Rankings.....	
Strengths.....	
Weaknesses.....	
Model 2 - Algorithm (Medians of Criteria).....	
Rankings.....	
Strengths.....	
Weaknesses.....	
Model 3 - Algorithm (Summation of Thrill Index Values).....	
Rankings.....	
Strengths.....	
Weaknesses.....	
Roller Coaster App.....	
Display.....	
Strengths.....	
Weaknesses.....	
Conclusions.....	
General Summary.....	
Analysis.....	
Future Extensions.....	
Predicting G-Forces for Missing Data Points.....	
Predicting Vertical Angles for Missing Data Points.....	
Base Weights Using Scientific Evidence.....	
Incorporate Artificial Intelligence into App.....	
References.....	
Appendix.....	
Appendix A.....	
Appendix B.....	
Appendix C.....	

Problem Statement + Introduction

The problem is that roller coaster ranking systems are often very subjective and do not carefully consider qualitative data that contributes to the ranking. Furthermore, there is a lack of apps that finds roller coasters for you based on your location and personal preferences.

There are three main things we need to do solve this problem:

1. Create an objective quantitative algorithm or set of algorithms to develop a descriptive roller coaster rating/ranking system based only on roller coaster numerical and descriptive specification data (e.g., speed, duration of ride, steel or wood, drop).
2. Use your algorithm(s) to develop your “Top 10 Roller Coasters in the World” list. Compare and discuss the rating/ranking results and descriptions from your team’s algorithm(s) with at least two other rating/ranking systems found online.
3. Describe the concept and design for a user-friendly app that uses your algorithm

Assumptions

We had to make multiple assumptions to create our models. The assumptions were necessary to determine which criteria were relevant to the algorithm and which criteria were relevant to the app. In addition, several assumptions were made to fill in missing data and account for special cases.

How Ranking is Determined?

After much deliberation, we decided that the ranking of a roller coaster is determinant on the experience of the rider, which mainly takes into account thrill. The better the rider's experience on the roller coaster, the better its ranking.

Parameter Relevance to Algorithm+App:

We were given many parameters of active roller coasters, such as height, speed, and G Force. However, not all of these were relevant to the algorithm and/or app. Thus, we had to determine which of these parameters were relevant and which ones could be neglected for the general algorithm and/or the app.

Location:

We decided that the park, city, state, country, and region were all negligible for the algorithm because we wanted our algorithm to be universal to people from all parts of the world. Thus, the algorithm we used to determine the top 10 best roller coasters in the world does not take location into consideration. However, because a person using the app may not be able to travel to distant areas to ride a certain roller coaster, location is still an important criteria for the app. Therefore, the app does take park, city, state, country, and region into consideration while generating recommendations and ratings.

Construction + Type:

Construction (steel or wooden) and type of roller coaster (flying, inverted, sit down, stand up, suspended, or wing) were neglected in the algorithm because we decided that construction and type of roller coaster were almost solely based on personal preference. We could not find any objective data online stating that steel roller coasters were better than wooden roller coasters, or vice versa. Each construction type of roller coaster had its own pros and cons, which balanced each other out. Likewise, we could not find any objective data online stating that one particular type of roller coaster was better than the others. However, because we came to consensus that construction and type of roller coaster were mainly based on personal preference, we decided that they would be a factor in determining the best rides for a specific person in our app.

Status:

The status of the roller coaster was not considered in our algorithm because we assumed that our algorithm was meant to be applied to active roller coasters. If a roller coaster was inactive, then there was no point to assign a thrill ranking to it since no one would ride it. As for the app, we decided that we would just exclude all of the inactive roller coasters from the app's database so that it would not matter.

Year Built:

The year the roller coaster was built was not factored into our algorithm because the year that a roller coaster was built does not directly affect the thrill of roller coaster. Of course, older roller coasters may be shorter or slower, but those effects are already accounted for in the other given parameters. Thus, the year built should not be a factor in our "thrill" rating. However, we agreed that the year a roller coaster was built may be important to some people. While some people may prefer newer roller coasters, others may prefer older, vintage roller coasters. Thus, we decided to still incorporate the year built into our app.

Height, Length, and Max Drop Height:

We decided that height, which we assumed to be the height from the ground to the peak of the roller coaster, should not be used for the general algorithm because the overall height of the roller coaster does not matter. As for length of the roller coaster, the only contribution that has towards the experience of the ride is the duration of the ride, which is already given separately, so we also omitted the length of the roller coaster in our general algorithm. However, the max drop height is definitely a factor in the algorithm since a greater drop height results in a longer period of zero gravity, which in turn activates the adrenaline response, meaning that the higher the drop height, the greater the "thrill". We also decided to only include drop height as an option within the app so as to not confuse the user with both drop height vs height and also length vs duration.

Speed, Inversions, G Force, Vertical Angle:

Speed, number of inversions (if there are any), and G force are all taken into consideration in the algorithm. All of these factors contribute to "thrill", so all of these are factored into our algorithm. Increasing speed increases "thrill" because faster speeds mean bigger adrenaline rushes. The number of inversions that a roller coaster has is important to our algorithm because more inversions means greater "thrill". Finally, G force is important to our algorithm because increasing those things increases the "thrill" a person feels. Speed, number of inversions, and G force are important in our app as well because different people have contrasting preferences. Thus, we incorporated all of these criteria into the app. However, vertical angles are not included in either the algorithm or the app because although increasing vertical angle up to 180° increases "thrill", there is nowhere near enough data for us to incorporate vertical angle.

Duration:

Duration is important to both the general algorithm and the app because longer rides in terms of time result in a more extended time of thrill for the roller coaster riders. Thus, longer durations will contribute to higher “thrill” scores on our algorithm. Within the app, the user can choose whether they prefer long-lasting rides, quick rides, or if they have no preference.

Summary Table:

The table below shows whether a specific criteria is taken into consideration for the algorithm or app. A green highlight indicates that the criteria is relevant for the algorithm/app, while a red highlight indicates that the criteria is not relevant for the algorithm/app.

	Park	City	State	Country	Region	Construction
Algorithm						
App						

	Type	Status	Year Opened	Height	Speed	Length
Algorithm		N/A				
App		N/A				

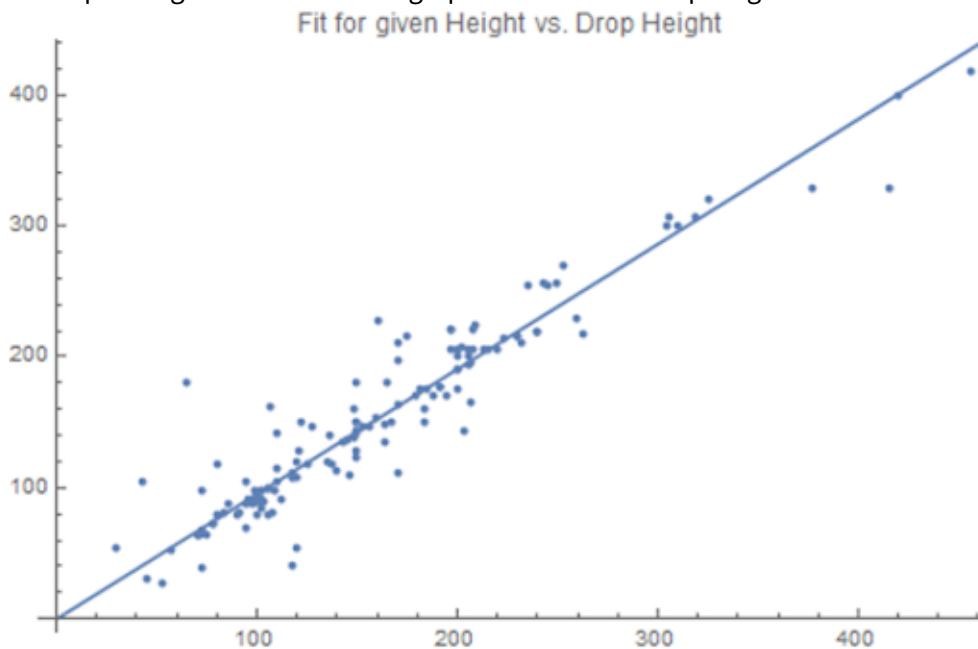
	Has inversions?	# of Inversions	Max Drop Height	Duration	G Force	Vertical Angle
Algorithm						
App						

Filling in Missing Data

The Excel spreadsheet we were given contained a lot of missing/unknown data. In order for us to be able to use an algorithm to rank roller coasters, we needed to fill in those gaps. There were missing heights, speeds, lengths, drop heights, durations, G-forces, and vertical angles. To figure out how to fill in the missing data, we hypothesized and looked for correlations with other existing criteria.

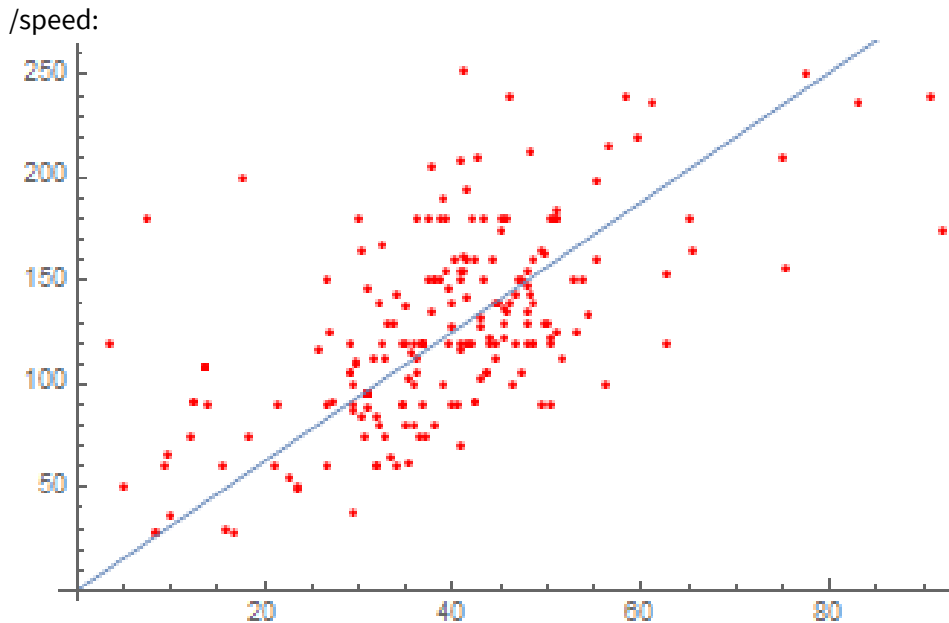
Heights and Drop Heights:

In order to find the missing heights, we assumed that height is proportionate to drop height. To find the constant of proportionality (k), we imported the existing data sets that contained both height (x) and drop height (y) into Mathematica. Drop height is a function of height. Then, we used “Fit” to draw a least squares linear regression line to fit the data. We also adjusted the y-intercept to 0 because using logical reasoning, we assumed that drop height should not be greater than height. Then, we took the slope of that regression line and used it as our constant of proportionality. Thus, drop height = $k \cdot \text{height}$, and rearranging the equation: height = drop height/ k . Finally, we simply substituted the drop heights into this equation to obtain the missing heights. We figured out that drop height = $k \cdot \text{height}$, so all we had to do was simply substitute height with the known heights to obtain the drop height of the corresponding roller coaster. The graph below shows drop height as a function of height:



Speeds, Lengths, and Durations:

Given that distance = speed \cdot time ($D = kst$) from basic physics principles, we can rearrange and modify this equation to get: $s = k \cdot (L/t)$, where s represents the speed of the roller coaster, L represents the length of the roller coaster, t represents the duration of the roller coaster, and k represents the constant of proportionality. To find the constant of proportionality (k), we imported the existing data sets that contained both length/speed (x) and duration (y) into Mathematica. Then, we used “Fit” to draw a least squares linear regression line to fit the data. We also adjusted the y-intercept to 0 because using logical reasoning, we assumed that duration cannot be greater than 0 seconds if the length and the speed are at 0 (since the ride hasn’t started yet). Then, we took the slope of that regression line and used it as our constant of proportionality. Now, we have the 3 variables needed for the equation: $s = k \cdot (L/t)$. The graph below shows duration as a function of length-



G-Forces:

Since there was a minimal quantity of G-Forces that were given to us and G-Forces were not directly related to any of the other criteria, we decided that we would fill in the missing G-Forces with the average G-Force in the original data, which was about 4.30 g's.

Vertical Angles:

Since the vertical angle measure was not given for most roller coasters and vertical angle was not directly related to any of the other criteria, we decided that we would not be using vertical angles for the algorithm at all.

Special Cases

In some cases, there were multiple criteria that were missing, so the formulas mentioned above would not work since there would be two variables in only one equation. Consequently, if this is such the case, we assume the criteria to be the average after filling in the rest of the data.

List of Assumptions

1. The factors that pertain to the algorithm are: speed, whether or not the roller coaster has inversions, number of inversions, max drop height, duration, and G force.
2. The factors that pertain to the app are: park, city, state, country, region, construction, type, year opened, height, speed, length, whether or not the roller coaster has inversions, number of inversions, max drop height, duration, G force, and vertical angle.
3. Drop height \propto height.

4. Duration is \propto length/velocity.
5. If there are multiple criteria missing for a certain roller coaster such that assumptions #3 and 4 would not work, we assume the criteria to be the average for that criteria.

Model 1 - Algorithm (Means of Criteria)

The first approach that our team took was to weigh each criteria equally to come up with one system that takes every one of those criteria into consideration. In order to ensure equal weighting, a roller coaster's data values of each statistic were divided by the average of that statistic as calculated from the given data of the 300 roller coasters. The rationale behind this is that the average, by definition, is a central or typical value of a data set. This means that the majority of the data should be centered around the average, so dividing each point the data set by the average of the dataset results in a range of values based around the 1. For all of the criteria used, a larger value in any criteria resulted in a more thrilling ride. Therefore, these values were summed up to obtain a base score, which is then compared to base scores of other roller coasters to create a ranking list.

Rankings:

	Source 2	Source 3	Model 1
#1	Kingda Ka	Millenium Force	Smiler
#2	Takabisha	Top Thrill Dragster	Steel Dragon 2000
#3	El Toro	Superman - Ride of Steel (Agawam, MA)	Medusa
#4	The Gravity Max	The Beast	Leviathan
#5	Superman - Ride of Steel (Agawam, MA)	Oblivion	Kingda Ka
#6	Formula Rossa	Superman - the Escape	Coaster Through the Clouds
#7	Top Thrill Dragster	Drachen Fire	Scream!
#8	Full Throttle	Goliath	Banshee
#9	The Smiler	Son of Beast	Alpengeist
#10	Hades 360	Superman - Ride of Steel (Corfu, MA)	Top Thrill Dragster

Strengths:

- Closest as we can get to 100% objective
- Every objective measurement is evenly balanced (no weighting)

Weaknesses:


- An outlier in as few as one measurement can offset the rankings drastically
- Some objective measures may contribute more towards the thrill of the ride than other ones.

- Assumes that there is no threshold value of thrill where thrill becomes extreme discomfort for the rider.

Model 2 - Algorithm (Medians of Criteria)

An alternative way to approach the problem is to apply the same logic as Approach 1, but each data point is divided by the median instead of the average. This works very similarly to the Approach 1, except it is less affected by outliers. However, the drawback of this alternative method is that the median is not always an accurate representation of the central value of the data, which could result in unanticipated weighting of the data.

Rankings:

	Source 2	Source 3	Model 2
#1	Kingda Ka	Millenium Force	Smiler 
#2	Takabisha	Top Thrill Dragster	Steel Dragon 2000
#3	El Toro	Superman - Ride of Steel (Agawam, MA)	Kingda Ka
#4	The Gravity Max	The Beast	Medusa
#5	Superman - Ride of Steel (Agawam, MA)	Oblivion	Leviathan
#6	Formula Rossa	Superman - the Escape	Top Thrill Dragster
#7	Top Thrill Dragster	Drachen Fire	Coaster Through the Clouds
#8	Full Throttle	Goliath	Banshee
#9	The Smiler	Son of Beast	Scream!
#10	Hades 360	Superman - Ride of Steel (Corfu, MA)	Alpengeist

Strengths:

- Very close to a purely objective approach
- Outliers have less impact on the base scores (and in turn the rankings)
- Almost evenly weighted criteria

Weaknesses:

- Does not account for the fact that some parameters may have a larger impact on the experience of the ride

- Model potentially has unexpected weighting due to the use of the median
- Assumes that there is no threshold value of thrill where thrill becomes extreme discomfort for the rider.

Model 3 (Algorithm): Summation of Thrill Index Values

The third model on an algorithm for ranking the roller coasters is based on the summation of thrill index values obtained from specific optimization functions of each specific roller coaster datum type (See Appendix C). Each optimization function was derived based on individual algorithm 'Parameter Relevance' criteria formulated by the team and is represented as a thrill index value as a function of the datum type. Each thrill index value is bounded between 0 and 1. The speed function is represented by a linear function where a higher speed yields a higher index value; the constant $1/149.1$ is formulated from the highest roller coaster speed value from the given list and constitutes to yielding an index value that is bounded between 0 and 1. The inversions function is represented by a linear function where a higher number of inversions yield a higher index value; the constant $1/14$ is formulated from the highest number of inversions from the given list and constitutes to yielding an index value that is bounded between 0 and 1. The G-force function is represented by a bell curve function where the curve's global maximum is represented by the best determined G-force from (See Ref. 1) and yields the highest index value. The drop distance function is represented by a linear function where a higher angle yields a higher index value; the constant $1/418$ is formulated from the highest drop height and constitutes to yielding an index value between 0 and 1. The duration function is represented by a linear function where a longer duration yields a higher index value; the constant $1/325$ is formulated from the longest duration from the give roller coaster data and constitutes to yielding an index value between 0 and 1.

Rankings:

	Source 2	Source 3	Model 3
#1	Kingda Ka	Millenium Force	Smiler
#2	Takabisha	Top Thrill Dragster	Steel Dragon 2000
#3	El Toro	Superman - Ride of Steel (Agawam, MA)	Hyperion
#4	The Gravity Max	The Beast	Medusa
#5	Superman - Ride of Steel (Agawam, MA)	Oblivion	Flashback
#6	Formula Rossa	Superman - the Escape	Leviathan
#7	Top Thrill Dragster	Drachen Fire	Kingda Ka
#8	Full Throttle	Goliath	Gao
#9	The Smiler	Son of Beast	Bullet Coaster
#10	Hades 360	Superman - Ride of Steel (Corfu, MA)	Intimidator

Strengths:

- Very close to completely objective, does not require any subjective data but the amount of functions was determined based on the team Parameter Relevance.
- Each function follows a logical pattern, which contributes to yielding accurate results: ex: speed is represented as a linear graph since the faster the roller coaster goes the more “thrilling” it is.
- Adaptable and flexible, can be dynamically implemented into the app where certain values, i.e. bell curve maximums, would change based on user input. Or hard-coded to be completely objective.

Weaknesses:

- Functions represent only a rough estimation of an accurate thrill index value.
- Requires all specification data types of roller coasters in order to yield an accurate summation of thrill index values.

Roller Coaster App

First, the app asks for the user’s general location and asks how far they are willing to travel to get to a theme park. He or she is given the option to select a theme park or the app will designate a circular region around the user based on the time the distance they inputted beforehand. The app stores this data as a preference of the user, which can be changed at any time in the settings. If there is no theme park in the region, the app will choose the nearest theme park to them.

The main browse menu of the app will then appear and ask the user if they are looking to just browse roller coasters in the region or if they would like help to find a rollercoaster that best suits their preferences. If the first option is chosen, the user is free to navigate the app and utilize the search and advanced search options. If the second option is chosen, the user will be shown a series of questions in the format of “Do you like [option]?” about their preferences to which they may answer “I don’t like [option]”, “I don’t have a preference on [option]”, or “I like [option].” The questions will be asking about the user’s opinions towards big drops (drop height), being pressed into their seat (g force), wooden or steel roller coasters (to which the user will have the option of wooden, steel, or no preference), long roller coasters (duration), and going upside down (inversions).

The app will then weight the different categories based on what the user entered as their preferences. For dislike, no preference, and like, -1, 0, or 1 would be the respective weights for the corresponding values of the roller coasters, and the app would then put these weights into the algorithm defined by model #2 and display the results of roller coasters within the given region that achieve the best score based on the preferences given by the user. The user is then free to select a rollercoaster and see a readout of specs on the rollercoaster and reviews from previous users.

The search bar may be used by the user to search for specific roller coasters. The user may also use the advanced search option to check different categories of coasters such as flying, sit down, inverted, suspended, or wing, which would then narrow down the results based on the preferences of the user. Then, the app will display a list of those roller coasters where the user can click into any roller coaster to find out more about its statistics, which will include max drop height, G-value, etc.

Display of App:

Hello! Enter your location so we can find the best roller coasters for you

How far are you willing to travel to get to a theme park?

_____miles

Do you want to:

Browse Roller Coasters

or

Get help finding the best roller coaster for you


Do you like big drops?

I do like big drops!


I have no preference

I don't like big drops

Your results:



1. Kingda Ka



2. Bizarro

Strengths:

- Closest as we can get to 100% objective
- Every objective measurement is evenly balanced (no weighting)

Weaknesses:

- An outlier in as few as one measurement can offset the rankings drastically
- Some objective measures may contribute more towards the thrill of the ride than other ones.
- Assumes that there is no threshold value of thrill where thrill becomes extreme discomfort for the rider.

Conclusions

General Summary:

After performing the tests, the results for the most “thrilling” roller coaster were successfully acquired. The results across all models have shown that the most “thrilling” roller coaster is “Smiler”, scoring a value of 7.578664751 on model #1, 8.253148665 on model #2, and 2.465103196 on model #3. Additionally, all approaches had similar “high-ranked” roller coasters in the top 10 placeholders; the list of these roller coasters includes “Steel Dragon 2000”, “Kingda Ka”, “Medusa”, and “Leviathan”, with values of 6.3 and higher in model #1, 6.8 and higher in model #2, and 2.1 and higher in model #3.

Conclusive Analysis

Based on the analysis made in ‘Comparisons’. The results of the models have similarities but are mostly different to the results of external sources. This shows that the results of the models are minorly favorable with the majority of the online sources, while all ranking systems contain some similar rides, ex: “Kingda Ka”, the highest rankings are different: “Smiler” for . This suggests that the model results yield mostly different outcomes while remaining mostly objective.

Model 1

Smiler	Alton Towers	Alton	Staffordshire, England	United Kingdom	Europe	7.5786648
Steel Dragon 2000	Nagashima Spa Land	Nagashima	Kuwana, Mie	Japan	Asia	6.93678
Medusa	Six Flags Discovery Kingdom	Vallejo	California	United States	North America	6.6651619
Leviathan	Canada's Wonderland	Vaughan	Ontario	Canada	North America	6.6161552
Kingda Ka	Six Flags Great Escape	Jackson	New Jersey	United States	North America	6.5961185
Coaster Through the Clouds	Nanchang Wanda Theme Park	Xinjian	Nanchang, Jiangxi	China	Asia	6.467795
Scream!	Six Flags Magic Mountain	Valencia	California	United States	North America	6.3910734
Banshee	Kings Island	Mason	Ohio	United States	North America	6.3790694
Alpengeist	Busch Gardens Williamsburg	Williamsburg	Virginia	United States	North America	6.3733875
Top Thrill Dragster	Cedar Point	Sandusky	Ohio	United States	North America	6.3399125
Intimidator 305	Kings Dominion	Doswell	Virginia	United States	North America	6.3052527
Fury 325	Carowinds	Charlotte	North Carolina	United States	North America	6.2858408
Soaring Dragon & Dancing Phoenix	Nanchang Wanda Theme Park	Xinjian	Nanchang, Jiangxi	China	Asia	6.2461819
GateKeeper	Cedar Point	Sandusky	Ohio	United States	North America	6.2182883

Model 2

Smiler	Alton Towers	Alton	Staffordshire, England	United Kingdom	Europe	8.253149
Steel Dragon 2000	Nagashima Spa Land	Nagashima	Kuwana, Mie	Japan	Asia	7.462539
Kingda Ka	Six Flags Great Escape	Jackson	New Jersey	United States	North America	7.287693
Medusa	Six Flags Discovery Kingdom	Vallejo	California	United States	North America	7.186111
Leviathan	Canada's Wonderland	Vaughan	Ontario	Canada	North America	7.135001
Top Thrill Dragster	Cedar Point	Sandusky	Ohio	United States	North America	6.999136
Coaster Through the Clouds	Nanchang Wanda Theme Park	Xinjian	Nanchang, Jiangxi	China	Asia	6.916293
Banshee	Kings Island	Mason	Ohio	United States	North America	6.900709
Scream!	Six Flags Magic Mountain	Valencia	California	United States	North America	6.896875
Alpengeist	Busch Gardens Williamsburg	Williamsburg	Virginia	United States	North America	6.888426
Fury 325	Carowinds	Charlotte	North Carolina	United States	North America	6.821743
Intimidator 305	Kings Dominion	Doswell	Virginia	United States	North America	6.811852
Soaring Dragon & Dancing Phoenix	Nanchang Wanda Theme Park	Xinjian	Nanchang, Jiangxi	China	Asia	6.796926
Dragon Khan	PortAventura Park	Salou	Tarragona	Spain	Europe	6.773489

Model 3

Name	Park	City/Region	City/State/Region	Country/Region	Geographic Region	App2 Index Value
Smiler	Alton Towers	Alton	Staffordshire, England	United Kingdom	Europe	2.465103196
Steel Dragon 2000	Nagashima Spa Land	Nagashima	Kuwana,, Mie	Japan	Asia	2.344441654
Hyperion	Energylandia	Zator	Malopolskie	Poland	Europe	2.274582747
Medusa	Six Flags Discovery Kingdom	Vallejo	California	United States	North America	2.262680143
Flashback	Six Flags New England	Agawam	Massachusetts	United States	North America	2.231012431
Leviathan	Canada's Wonderland	Vaughan	Ontario	Canada	North America	2.223945516
Kingda Ka	Six Flags Great Escape	Jackson	New Jersey	United States	North America	2.179490638
Gao	Greenland	Arao	Kumamoto	Japan	Asia	2.135007343
Bullet Coaster	Happy Valley	Nanshan	Shenzhen, Guangdong	China	Asia	2.118413972
Intimidator 305	Kings Dominion	Doswell	Virginia	United States	North America	2.110023787
Top Thrill Dragster	Cedar Point	Sandusky	Ohio	United States	North America	2.088927018
Titan	Six Flags Over Texas	Arlington	Texas	United States	North America	2.061141436
Scream	Six Flags Magic Mountain	Valencia	California	United States	North America	2.048554492

Future Extensions

Predicting G-Forces for Missing Data Points

Unfortunately, we were given very minimal G-force data on the roller coasters. In addition, it was difficult to calculate G-force or even to approximate G-force, because in order to estimate G-force, we would need the average acceleration of the roller coaster, which was not given to us. Thus, we would need to calculate the accelerations of each of the roller coasters in order to calculate the G-force, and we could not find any online resources that described how to find acceleration from the specific data points we were given. While it would not be impossible to calculate G-force, it would be very time-consuming and challenging since an in-depth understanding of calculus is needed. If our group had more time and knew advanced calculus, we could better estimate the G-forces rather than assuming that missing G-forces would be the same as the average G-force.

Predicting Vertical Angles for Missing Data Points

Similar to G-force, we were given very minimal vertical angle data on the roller coasters. There was even less data on vertical angles than on G-forces. In addition, it would be nearly impossible for us to calculate vertical angles because there are no clear correlations between vertical angles with the other parameters. In addition, we could not find anything online that suggested that vertical angle is related to speed, drop height, G force, etc. Thus, we decided to not include vertical angle in either the algorithm or the app, but vertical angle still remains a major factor in “thrill” rating. Thus, as a possible extension, we would attempt to predict vertical angle based on other factors, and incorporate vertical angles into both our algorithm and app.

Base Weights Using Scientific Evidence

It is logically impossible to base parameters completely objectively, because any sort of weighted system will be at least a little subjective. We did not find any scientific evidence online which suggested that some parameters should be more heavily weighted than others, so we determined how much to weight each parameter in both the algorithm and the app based on our decisions as a group, which still remains subjective. If we were to find scientific evidence that suggests some parameters contribute more to “thrill” than other parameters, we could create less subjective weighted rankings.

Incorporate Artificial Intelligence into App

We could incorporate artificial intelligence into our app to add additional features and make it for functional. For instance, we could match two or more people with similar roller coaster preferences living in the same area so that they could potentially ride together on some roller coasters. In addition, our app could automatically be updated when a roller coaster gets shut down or closed.

References

1. "Coasterpedia The Roller Coaster Wiki." Coasterpedia, coasterpedia.net/.
2. Delatte, Thomas. "The Fifteen Most Thrilling Roller Coasters in the World." Gunaxin Travel, 10 July 2017, travel.gunaxin.com/15-thrilling-roller-coasters-world/197186.
3. Dynn, www.eng.buffalo.edu/~hulme/topcoast.html.
4. "G-Forces." GO FLIGHT MEDICINE, goflightmedicine.com/pulling-gs/.
5. "Random Roller Coaster." Expedition Everest - Walt Disney World - Disney's Animal Kingdom (Lake Buena Vista, Florida, USA), rcdb.com/.
6. "Ultimate Rollercoaster." Coney Island | Roller Coaster History, www.ultimaterollercoaster.com/.

Appendix

Appendix A: Filling in Durations, Lengths, and Speeds

```

In[ ]:= x1 = {42.25454545`, 15.43920803`, 32.50733138`, 38.95522388`, 48.03719008`,
48.02000325`, 35.95041322`, 39.15351116`, 45.59775841`, 17.55304555`,
33.40909091`, 42.81793788`, 49.46542894`, 41.35026738`, 31.44385027`, 32.23140496`,
36.81818182`, 36.72272727`, 36.81818182`, 36.72272727`, 36.81818182`, 31.0110664`,
37.0431681`, 45.54545455`, 77.43055556`, 48.61261872`, 50.6480344`, 83.05524302`,
65.04058442`, 44.54172876`, 38.03798858`, 35.9527972`, 43.60227273`, 13.56382979`,
13.56382979`, 13.56382979`, 13.56382979`, 13.56382979`, 13.56382979`,
13.56382979`, 13.56382979`, 13.56382979`, 53.69318182`, 26.63961039`, 41.23376623`,
62.74922261`, 36.13636364`, 41.19150081`, 92.06737796`, 42.25454545`, 32.66820432`,
29.11931818`, 26.59090909`, 34.7107438`, 29.04545455`, 29.04545455`, 31.0110664`,
23.41458059`, 49.79829545`, 45.01704545`, 43.68881119`, 75, 32.93974225`,
42.85714286`, 55.2912503`, 36.58237875`, 31.73981191`, 45.54545455`, 31.0110664`,
39.82676414`, 13.56382979`, 30.81818182`, 31.0110664`, 34.1540404`, 34.1540404`,
26.90082645`, 29.34518452`, 56.61031728`, 21.42857143`, 22.66334381`,
61.10277653`, 42.37449118`, 44.71590909`, 55.19074675`, 32.72727273`, 36.09625668`,
12.51385809`, 36.72272727`, 37.94805195`, 34.93636364`, 29.84635083`, 40.2892562`,
3.60031348`, 34.79665072`, 49.74527888`, 41.1406518`, 37.65264586`, 75.27272727`,
48.32727273`, 38.63636364`, 13.82454545`, 47.72727273`, 18.28349282`, 40.45454545`,
50.27936481`, 62.79066986`, 41.55291303`, 31.98605372`, 16.60866477`, 31.0110664`,
43.81468531`, 45.20454545`, 46.71032357`, 40.65711462`, 31.0110664`, 48.35227273`,
50.90909091`, 44.397463`, 40.89691558`, 43.34415584`, 41.2972028`, 48.35043988`,
31.0110664`, 31.0110664`, 48.02000325`, 9.917355372`, 45.26136364`, 31.0110664`,
42.0025729`, 32.03181818`, 35.09937238`, 33.75902611`, 42.8553719`, 41.74397032`,
34.7107438`, 45.97159091`, 12.2526738`, 29.44852941`, 29.71407625`, 35.01136364`,
40.01626281`, 25.6684492`, 48.48484848`, 31.98605372`, 30.73863636`, 40.93572443`,
44.45103206`, 56.11550285`, 43.932247`, 47.23225405`, 31.0110664`, 45.33492823`,
39.77272727`, 34.68899522`, 41.37426901`, 32.48863636`, 31.0110664`, 45.83916084`,
50.4358655`, 46.75324675`, 40.70574163`, 44.31818182`, 29.28099174`, 43.12770563`,
35.45454545`, 42.0001637`, 40.90909091`, 9.621212121`, 38.73966942`, 45.92983063`,
9.490469208`, 5.139217597`, 49.56869835`, 27.35717834`, 30.17111267`, 47.97769924`,
20.99012917`, 46.34881423`, 50.84194215`, 58.3722488`, 38.81118881`, 50.90909091`,
52.88697789`, 35.97902098`, 23.63636364`, 12.51385809`, 49.96886675`, 37.40641711`,
39.62340799`, 50.97550573`, 39.10714286`, 47.81582054`, 8.420454545`, 30.41866029`,
36.02108037`, 29.55925325`, 90.87055678`, 7.381313131`, 29.53420342`, 54.41680961`,
40.93620098`, 42.60962567`, 53.02447552`, 15.90909091`, 8.420454545`, 59.68899522`,
38.0324989`, 47.15454545`, 37.30519481`, 47.10743802`, 65.55630936`, 50.32467532`,
51.45797599`, 50.30712531`, 26.67613636`, 39.57692308`, 37.82279315`, 35.37039701` }

```

```
In[ ]:= y1 = {92, 60, 167, 190, 143, 130, 120, 180, 135, 200, 64, 128, 90, 160, 112, 140, 120, 90, 120,
120, 120, 96, 75, 122, 250, 160, 180, 236, 180, 140, 150, 80, 105, 108, 108, 108, 108,
108, 108, 108, 108, 108, 108, 150, 150, 162, 120, 105, 252, 175, 92, 75, 120, 90, 120,
105, 105, 96, 49, 163, 180, 105, 210, 130, 102, 198, 75, 85, 130, 96, 128, 108, 146,
88, 144, 60, 125, 38, 216, 90, 55, 236, 160, 140, 160, 112, 180, 92, 120, 80, 120, 180,
160, 120, 80, 130, 155, 135, 156, 213, 180, 90, 120, 75, 90, 90, 154, 142, 60, 28, 96,
122, 174, 120, 208, 96, 120, 180, 120, 155, 150, 195, 140, 96, 96, 148, 36, 180, 96,
120, 80, 103, 130, 132, 120, 90, 140, 75, 87, 111, 138, 140, 117, 120, 61, 75, 150,
113, 100, 120, 150, 96, 136, 90, 90, 120, 120, 96, 180, 122, 144, 70, 160, 100, 180,
116, 180, 120, 66, 150, 240, 60, 50, 165, 92, 165, 135, 60, 100, 125, 240, 100, 180,
150, 100, 50, 92, 130, 180, 146, 184, 155, 155, 28, 85, 112, 110, 240, 180, 90, 133,
117, 210, 125, 30, 28, 220, 150, 105, 150, 150, 165, 120, 112, 180, 60, 120, 205, 62}
```

```
In[ ]:= data = Transpose[{x1, y1}];
```

```
In[ ]:= fit[x_] = Fit[data, {x}, x]
```

```
Set: Tag Plus in (62.9344 + 0.0196676 x)[x_] is Protected.
```

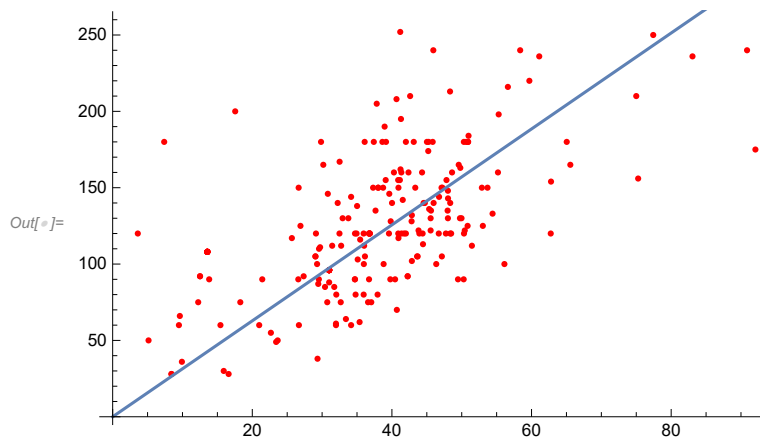
```
Out[ ]:= 3.1412 x
```

```
In[ ]:= dataPlot = ListPlot[data, PlotStyle -> Red]
```

```
In[ ]:= fitPlot = Plot[3.141203830056361 x, {x, 0, 10000}]
```

```
In[ ]:= Clear[x]
```

```
In[ ]:= Show[dataPlot, fitPlot]
```



Appendix A: Filling in Heights and Drop Heights

Variables

- height: List of all given heights
- heightDrop: List of all heights with given drops with zeroes for holes
- heightDropSkips: List of all heights with given drops skipping holes
- heightNoDrop: List of all heights without given drops with zeroes for holes
- drop: List of all given drops with zeroes for holes

- dropSkips: List of all given drops skipping holes
- fit[x]: Equation for line of best fit
- dropFill: List of all drops, given and predicted
- heightDropFitList: Transposed heightDropSkips and dropSkips

Inputting Data

Java was used to remove excel formatting and make it easier to import into Mathematica

```

1 import java.util.Scanner;
2 public class HiMCM {
3     public static void main(String[] Args)
4     {
5         Scanner scanny = new Scanner(System.in);
6         scanny.close();
7         String data = "";
8         data = data.replaceAll("\\\\r\\n", ", ");
9         data = data.trim();
10        System.out.println(data);
11    }
12 }

```

```

In[ ]:= height = {98.4, 151.6, 72, 113, 195, 108.3, 108.3, 127, 118.1, 100, 95, 170, 98.4, 45.2, 118.1,
91.2, 167, 78, 117.8, 100, 105, 105, 105, 105, 109.3, 105, 115, 110, 230, 203, 98.4,
213, 72.2, 104, 142, 85.3, 124.7, 122, 78, 96, 64, 116.5, 116.5, 116.5, 116.5, 116.5,
116.5, 116.5, 116.5, 116.5, 116.5, 122, 110, 196.8, 208, 48, 120, 55, 242.8, 85.3, 98.4,
78.7, 85, 85, 98.4, 108.3, 80, 102, 103.7, 109.3, 62.3, 209, 230, 226.4, 59.1, 160.8,
148, 186, 150, 249.3, 181, 80.5, 83.7, 173.9, 170.6, 121, 115, 109.3, 200.2, 116.5,
102, 109.3, 74.2, 74.2, 101.7, 131.3, 115, 206.7, 72.2, 170.6, 259.2, 160, 46, 325,
131.3, 170, 125.3, 118, 70, 73, 235, 200, 165, 191.6, 105, 118, 108, 108.3, 205, 91,
136, 98.4, , 57.4, 134.5, 95, 200.2, 252.6, 110, 120, 232, 305, 131.3, 76, 179, 180.4,
100, 120, 95, 100, 71, 111.6, 138, 137.8, 164.1, 101.7, 456, 109.3, 149, 143, 99, 306,
207, 109.3, 205, 200, 205, 98.4, 30, 140, 105, 150, 98, 310, 109.3, 109.3, 83, 131.3,
110, 36.1, 70, 148, 139.1, 150, 109.3, 218, 218, 109.2, 42.6, 95.2, 85.3, 113, 153, 60,
230, 65, 196.8, 107, 48, 53, 149, 147.7, 160, 78, 101.7, 78.8, 102.3, 147.7, 164.1, 111,
88, 202, 106, 63, 109.3, 137, 80, 80, 367.3, 97.5, 44.4, 109.3, 156, 208, 73, 80, 145,
100, 239.5, 150, 90, 249.3, 116, 56, 146, 239.5, 150, 151.6, 200, 72.2, 117.8, 196.8,
196.8, 65.6, 64, 84, 75, 105, 205.1, 318.3, 150, 200, 205, 185, 100, 150, 191, 64, 197,
106, 164.1, 219.8, 168, 208, 415, 127, 183.8, 141, 135, 98.4, 170, 45.9, 38.4, 29, 80,
262.5, 96, 100, 75, 100, 245, 75, 98, 420, 64, 377.3, 121, 100, 107, 150, 223, 124, 108,
185, 100, 188, 148, 159, 64, 70, 109, 215, 98, 207, 75, 183.8, 57.1, 131.3, 175, 145.3};

```



```

In[ ]:= heightDropSkips = {195, 127, 118.1, 100, 95, 170, 45.2, 91.2, 167, 110, 203, 213, 72.2, 85.3,
    78, 96, 122, 110, 196.8, 120, 242.8, 98.4, 80, 102, 103.7, 209, 230, 148, 181, 170.6,
    200.2, 102, 259.2, 325, 170, 125.3, 118, 70, 235, 200, 165, 191.6, 105, 118, 108, 136,
    95, 200.2, 252.6, 110, 120, 232, 305, 179, 120, 95, 71, 111.6, 164.1, 101.7, 456,
    149, 143, 306, 207, 205, 200, 205, 30, 140, 105, 150, 310, 83, 150, 109.2, 42.6, 153,
    230, 65, 196.8, 107, 53, 149, 160, 78, 101.7, 102.3, 164.1, 202, 137, 80, 97.5, 156,
    208, 73, 145, 239.5, 150, 90, 249.3, 146, 239.5, 72.2, 117.8, 318.3, 150, 200, 205,
    185, 150, 191, 197, 106, 219.8, 208, 415, 183.8, 135, 170, 80, 262.5, 100, 100, 245,
    420, 377.3, 121, 223, 100, 188, 148, 159, 70, 215, 98, 207, 75, 183.8, 57.1, 175};

In[ ]:= dropSkips = {170, 147, 40, 90, 87.3, 210, 31.2, 81.7, 150, 141, 144, 205, 39.3, 88.6, 72, 92,
    150, 115, 221.2, 120, 255.9, 98, 80, 90, 90, 225, 215, 161, 176, 196.8, 190.3, 91,
    229.7, 320, 164, 118, 108, 65, 255, 175, 180, 177, 80, 108, 81.2, 140, 105, 190.3,
    269, 104.9, 108, 211, 300, 171, 54, 70, 65, 91.8, 147.7, 98.4, 418, 144, 135, 306,
    165, 194.7, 200, 205, 54, 113, 100, 150, 300, 80.5, 128, 98, 104, 147, 215, 180,
    221.2, 162, 27, 123, 228, 72, 98.4, 84.9, 134.5, 208, 119, 118, 91.4, 146, 205,
    67, 137, 219.8, 141, 80, 255.9, 109, 219.8, 98.4, 111.8, 306.8, 150, 205, 200,
    175, 180, 177, 205, 100, 205, 221, 328.1, 150.9, 120, 111, 80, 218.2, 95, 90, 255,
    400, 328.1, 128, 214, 80, 171, 138, 154, 64, 206, 88, 196, 64, 160.8, 52.5, 215};

In[ ]:= heightDrop = {0, 0, 0, 0, 195, 0, 0, 127, 118.1, 100, 95, 170, 0, 45.2, 0, 91.2, 167, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 110, 0, 203, 0, 213, 72.2, 0, 0, 85.3, 0, 0, 78, 96, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 122, 110, 196.8, 0, 0, 120, 0, 242.8, 0, 0, 0, 0, 0, 98.4, 0, 80, 102,
    103.7, 0, 0, 209, 230, 0, 0, 0, 148, 0, 0, 0, 181, 0, 0, 0, 170.6, 0, 0, 0, 200.2, 0, 102,
    0, 0, 0, 0, 0, 0, 0, 0, 259.2, 0, 0, 325, 0, 170, 125.3, 118, 70, 0, 235, 200, 165,
    191.6, 105, 118, 108, 0, 0, 0, 136, 0, 0, , 0, 95, 200.2, 252.6, 110, 120, 232, 305, 0,
    0, 179, 0, 0, 120, 95, 0, 71, 111.6, 0, 0, 164.1, 101.7, 456, 0, 149, 143, 0, 306, 207,
    0, 205, 200, 205, 0, 30, 140, 105, 150, 0, 310, 0, 0, 83, 0, 0, 0, 0, 0, 0, 150, 0, 0,
    0, 109.2, 42.6, 0, 0, 0, 153, 0, 230, 65, 196.8, 107, 0, 53, 149, 0, 160, 78, 101.7, 0,
    102.3, 0, 164.1, 0, 0, 202, 0, 0, 0, 137, 0, 80, 0, 97.5, 0, 0, 156, 208, 73, 0, 145, 0,
    239.5, 150, 90, 249.3, 0, 0, 146, 239.5, 0, 0, 0, 72.2, 117.8, 0, 0, 0, 0, 0, 0, 0, 0,
    318.3, 150, 200, 205, 185, 0, 150, 191, 0, 197, 106, 0, 219.8, 0, 208, 415, 0, 183.8, 0,
    135, 0, 170, 0, 0, 0, 80, 262.5, 0, 100, 0, 100, 245, 0, 0, 420, 0, 377.3, 121, 0, 0, 0,
    223, 0, 0, 0, 100, 188, 148, 159, 0, 70, 0, 215, 98, 207, 75, 183.8, 57.1, 0, 175, 0};

In[ ]:= drop = {0, 0, 0, 0, 170, 0, 0, 147, 40, 90, 87.3, 210, 0, 31.2, 0, 81.7, 150, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 141, 0, 144, 0, 205, 39.3, 0, 0, 88.6, 0, 0, 72, 92, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 150, 115, 221.2, 0, 0, 120, 0, 255.9, 0, 0, 0, 0, 98, 0, 80, 90, 90, 0, 0,
    225, 215, 0, 0, 0, 161, 0, 0, 0, 176, 0, 0, 0, 196.8, 0, 0, 0, 190.3, 0, 91, 0, 0, 0,
    0, 0, 0, 0, 0, 229.7, 0, 0, 320, 0, 164, 118, 108, 65, 0, 255, 175, 180, 177, 80,
    108, 81.2, 0, 0, 0, 140, 0, 0, 0, 105, 190.3, 269, 104.9, 108, 211, 300, 0, 0, 171, 0,
    0, 54, 70, 0, 65, 91.8, 0, 0, 147.7, 98.4, 418, 0, 144, 135, 0, 306, 165, 0, 194.7,
    200, 205, 0, 54, 113, 100, 150, 0, 300, 0, 0, 80.5, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 98,
    104, 0, 0, 0, 147, 0, 215, 180, 221.2, 162, 0, 27, 123, 0, 228, 72, 98.4, 0, 84.9, 0,
    134.5, 0, 0, 208, 0, 0, 0, 119, 0, 118, 0, 91.4, 0, 0, 146, 205, 67, 0, 137, 0, 219.8,
    141, 80, 255.9, 0, 0, 109, 219.8, 0, 0, 0, 98.4, 111.8, 0, 0, 0, 0, 0, 0, 0, 306.8,
    150, 205, 200, 175, 0, 180, 177, 0, 205, 100, 0, 205, 0, 221, 328.1, 0, 150.9, 0,
    120, 0, 111, 0, 0, 0, 80, 218.2, 0, 95, 0, 90, 255, 0, 0, 400, 0, 328.1, 128, 0, 0, 0,
    214, 0, 0, 0, 80, 171, 138, 154, 0, 64, 0, 206, 88, 196, 64, 160.8, 52.5, 0, 215, 0};

```

```
In[ ]:= heightNoDrop = {98.4, 151.6, 72, 113, 0, 108.3, 108.3, 0, 0, 0, 0, 98.4, 0, 118.1, 0, 0, 78,
  117.8, 100, 105, 105, 105, 105, 109.3, 105, 115, 0, 230, 0, 98.4, 0, 0, 104, 142, 0, 124.7,
  122, 0, 0, 64, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5, 116.5,
  0, 0, 0, 208, 48, 0, 55, 0, 85.3, 98.4, 78.7, 85, 85, 0, 108.3, 0, 0, 0, 109.3, 62.3, 0,
  0, 226.4, 59.1, 160.8, 0, 186, 150, 249.3, 0, 80.5, 83.7, 173.9, 0, 121, 115, 109.3,
  0, 116.5, 0, 109.3, 74.2, 74.2, 101.7, 131.3, 115, 206.7, 72.2, 170.6, 0, 160, 46, 0,
  131.3, 0, 0, 0, 0, 73, 0, 0, 0, 0, 0, 0, 108.3, 205, 91, 0, 98.4, 57.4, 134.5, 0, 0,
  0, 0, 0, 0, 131.3, 76, 0, 180.4, 100, 0, 0, 100, 0, 0, 138, 137.8, 0, 0, 0, 109.3, 0,
  0, 99, 0, 0, 109.3, 0, 0, 0, 98.4, 0, 0, 0, 0, 98, 0, 109.3, 109.3, 0, 131.3, 110, 36.1,
  70, 148, 139.1, 0, 109.3, 218, 218, 0, 0, 95.2, 85.3, 113, 0, 60, 0, 0, 0, 0, 48, 0, 0,
  147.7, 0, 0, 0, 78.8, 0, 147.7, 0, 111, 88, 0, 106, 63, 109.3, 0, 80, 0, 367.3, 0, 44.4,
  109.3, 0, 0, 0, 80, 0, 100, 0, 0, 0, 0, 116, 56, 0, 0, 150, 151.6, 200, 0, 0, 196.8, 196.8,
  65.6, 64, 84, 75, 105, 205.1, 0, 0, 0, 0, 0, 100, 0, 0, 64, 0, 0, 164.1, 0, 168, 0, 0,
  127, 0, 141, 0, 98.4, 0, 45.9, 38.4, 29, 0, 0, 96, 0, 75, 0, 0, 75, 98, 0, 64, 0, 0, 100,
  107, 150, 0, 124, 108, 185, 0, 0, 0, 0, 64, 0, 109, 0, 0, 0, 0, 0, 131.3, 0, 145.3};
```

```
In[ ]:= heightDropFitList = Transpose[{heightDropSkips, dropSkips}];
```

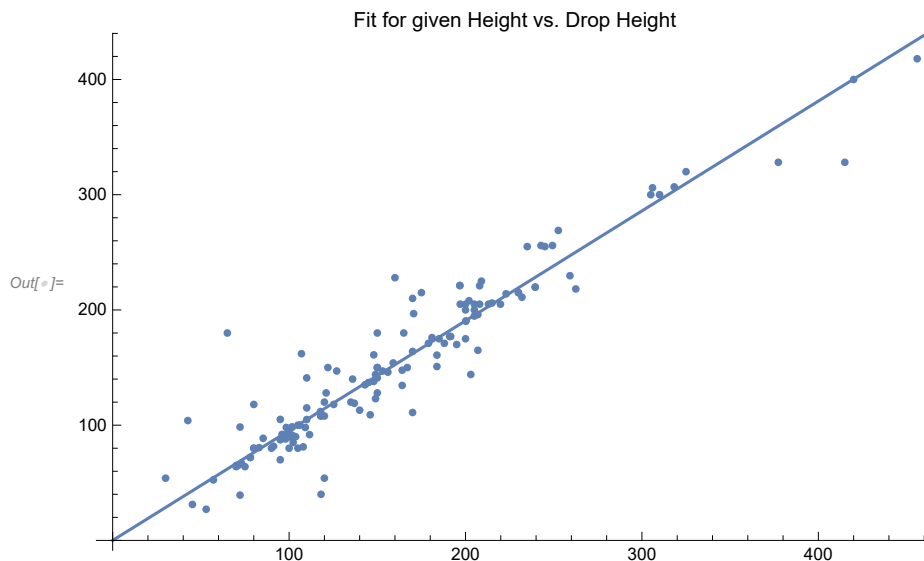
```
In[ ]:= fit[x_] = Fit[heightDropFitList, {x}, x]
```

```
Out[ ]:= 0.953096 x
```

```
In[ ]:= dropFill = fit[heightNoDrop];
```

```
In[ ]:= fitPlot = Plot[fit[x], {x, 0, 1000}];
```

```
In[ ]:= Show[ListPlot[heightDropFitList], fitPlot,
  PlotLabel -> "Fit for given Height vs. Drop Height"]
```



dropFill is the list of all given and predicted drop heights to be used in future models.

$In[]:=$ **dropFill = dropFill + drop**

$Out[]:=$ {93.7846, 144.489, 68.6229, 107.7, 170., 103.22, 103.22, 147., 40., 90., 87.3, 210., 93.7846, 31.2, 112.561, 81.7, 150., 74.3415, 112.275, 95.3096, 100.075, 100.075, 100.075, 100.075, 104.173, 100.075, 109.606, 141., 219.212, 144., 93.7846, 205., 39.3, 99.122, 135.34, 88.6, 118.851, 116.278, 72., 92., 60.9981, 111.036, 111.036, 111.036, 111.036, 111.036, 111.036, 111.036, 111.036, 150., 115., 221.2, 198.244, 45.7486, 120., 52.4203, 255.9, 81.2991, 93.7846, 75.0086, 81.0132, 81.0132, 98., 103.22, 80., 90., 90., 104.173, 59.3779, 225., 215., 215.781, 56.328, 153.258, 161., 177.276, 142.964, 237.607, 176., 76.7242, 79.7741, 165.743, 196.8, 115.325, 109.606, 104.173, 190.3, 111.036, 91., 104.173, 70.7197, 70.7197, 96.9299, 125.141, 109.606, 197.005, 68.8135, 162.598, 229.7, 152.495, 43.8424, 320., 125.141, 164., 118., 108., 65., 69.576, 255., 175., 180., 177., 80., 108., 81.2, 103.22, 195.385, 86.7317, 140., 93.7846, 54.7077, 128.191, 105., 190.3, 269., 104.9, 108., 211., 300., 125.141, 72.4353, 171., 171.938, 95.3096, 54., 70., 95.3096, 65., 91.8, 131.527, 131.337, 147.7, 98.4, 418., 104.173, 144., 135., 94.3565, 306., 165., 104.173, 194.7, 200., 205., 93.7846, 54., 113., 100., 150., 93.4034, 300., 104.173, 104.173, 80.5, 125.141, 104.841, 34.4068, 66.7167, 141.058, 132.576, 128., 104.173, 207.775, 207.775, 98., 104., 90.7347, 81.2991, 107.7, 147., 57.1858, 215., 180., 221.2, 162., 45.7486, 27., 123., 140.772, 228., 72., 98.4, 75.104, 84.9, 140.772, 134.5, 105.794, 83.8724, 208., 101.028, 60.045, 104.173, 119., 76.2477, 118., 350.072, 91.4, 42.3175, 104.173, 146., 205., 67., 76.2477, 137., 95.3096, 219.8, 141., 80., 255.9, 110.559, 53.3734, 109., 219.8, 142.964, 144.489, 190.619, 98.4, 111.8, 187.569, 187.569, 62.5231, 60.9981, 80.0601, 71.4822, 100.075, 195.48, 306.8, 150., 205., 200., 175., 95.3096, 180., 177., 60.9981, 205., 100., 156.403, 205., 160.12, 221., 328.1, 121.043, 150.9, 134.387, 120., 93.7846, 111., 43.7471, 36.5989, 27.6398, 80., 218.2, 91.4972, 95., 71.4822, 90., 255., 71.4822, 93.4034, 400., 60.9981, 328.1, 128., 95.3096, 101.981, 142.964, 214., 118.184, 102.934, 176.323, 80., 171., 138., 154., 60.9981, 64., 103.887, 206., 88., 196., 64., 160.8, 52.5, 125.141, 215., 138.485}

Appendix B: Filling in Durations, Lengths, and Speeds

Variables

- lengthSpeed: List of length/speed values with corresponding given durations
- durationSkips: List of all given durations with missing values skipped
- durationFill: List of all durations, given and predicted
- lengthDurFitList: Transposed lengthSpeed and durationSkips

Inputting Data

```

In[ ]:= lengthSpeed = {42.25454545`, 15.43920803`, 32.50733138`, 38.95522388`, 48.03719008`,
    48.02000325`, 35.95041322`, 39.15351116`, 45.59775841`, 17.55304555`, 33.40909091`,
    42.81793788`, 49.46542894`, 41.35026738`, 31.44385027`, 32.23140496`, 36.81818182`,
    36.72272727`, 36.81818182`, 36.72272727`, 36.81818182`, 31.0110664`, 37.0431681`,
    45.54545455`, 77.43055556`, 48.61261872`, 50.6480344`, 83.05524302`, 65.04058442`,
    44.54172876`, 38.03798858`, 35.9527972`, 43.60227273`, 13.56382979`, 13.56382979`,
    13.56382979`, 13.56382979`, 13.56382979`, 13.56382979`, 13.56382979`, 53.69318182`,
    26.63961039`, 41.23376623`, 62.74922261`, 36.13636364`, 41.19150081`,
    92.06737796`, 42.25454545`, 32.66820432`, 29.11931818`, 26.59090909`,
    34.7107438`, 29.04545455`, 29.04545455`, 31.0110664`, 23.41458059`,
    49.79829545`, 45.01704545`, 43.68881119`, 75, 32.93974225`, 42.85714286`,
    55.2912503`, 36.58237875`, 31.73981191`, 45.54545455`, 31.0110664`, 39.82676414`,
    13.56382979`, 30.81818182`, 31.0110664`, 34.1540404`, 34.1540404`, 26.90082645`,
    29.34518452`, 56.61031728`, 21.42857143`, 22.66334381`, 61.10277653`, 42.37449118`,
    44.71590909`, 55.19074675`, 32.72727273`, 36.09625668`, 12.51385809`, 36.72272727`,
    37.94805195`, 34.93636364`, 29.84635083`, 40.2892562`, 3.60031348`, 34.79665072`,
    49.74527888`, 41.1406518`, 37.65264586`, 75.27272727`, 48.32727273`, 38.63636364`,
    13.82454545`, 47.72727273`, 18.28349282`, 40.45454545`, 50.27936481`, 62.79066986`,
    41.55291303`, 31.98605372`, 16.60866477`, 31.0110664`, 43.81468531`, 45.20454545`,
    46.71032357`, 40.65711462`, 31.0110664`, 48.35227273`, 50.90909091`, 44.397463`,
    40.89691558`, 43.34415584`, 41.2972028`, 48.35043988`, 31.0110664`, 31.0110664`,
    48.02000325`, 9.917355372`, 45.26136364`, 31.0110664`, 42.0025729`, 32.03181818`,
    35.09937238`, 33.75902611`, 42.8553719`, 41.74397032`, 34.7107438`, 45.97159091`,
    12.2526738`, 29.44852941`, 29.71407625`, 35.01136364`, 40.01626281`, 25.6684492`,
    48.48484848`, 31.98605372`, 30.73863636`, 40.93572443`, 44.45103206`, 56.11550285`,
    43.932247`, 47.23225405`, 31.0110664`, 45.33492823`, 39.77272727`, 34.68899522`,
    41.37426901`, 32.48863636`, 31.0110664`, 45.83916084`, 50.4358655`, 46.75324675`,
    40.70574163`, 44.31818182`, 29.28099174`, 43.12770563`, 35.45454545`, 42.0001637`,
    40.90909091`, 9.621212121`, 38.73966942`, 45.92983063`, 9.490469208`, 5.139217597`,
    49.56869835`, 27.35717834`, 30.17111267`, 47.97769924`, 20.99012917`, 46.34881423`,
    50.84194215`, 58.3722488`, 38.81118881`, 50.90909091`, 52.88697789`, 35.97902098`,
    23.63636364`, 12.51385809`, 49.96886675`, 37.40641711`, 39.62340799`, 50.97550573`,
    39.10714286`, 47.81582054`, 8.420454545`, 30.41866029`, 36.02108037`, 29.55925325`,
    90.87055678`, 7.381313131`, 29.53420342`, 54.41680961`, 40.93620098`, 42.60962567`,
    53.02447552`, 15.90909091`, 8.420454545`, 59.68899522`, 38.0324989`, 47.15454545`,
    37.30519481`, 47.10743802`, 65.55630936`, 50.32467532`, 51.45797599`,
    50.30712531`, 26.67613636`, 39.57692308`, 37.82279315`, 35.37039701` };

```

```
In[ ]:= durationSkips = {92, 60, 167, 190, 143, 130, 120, 180, 135, 200, 64, 128, 90, 160, 112, 140, 120,
    90, 120, 120, 120, 96, 75, 122, 250, 160, 180, 236, 180, 140, 150, 80, 105, 108, 108,
    108, 108, 108, 108, 108, 108, 108, 108, 150, 150, 162, 120, 105, 252, 175, 92, 75, 120,
    90, 120, 105, 105, 96, 49, 163, 180, 105, 210, 130, 102, 198, 75, 85, 130, 96, 128, 108,
    146, 88, 144, 60, 125, 38, 216, 90, 55, 236, 160, 140, 160, 112, 180, 92, 120, 80, 120,
    180, 160, 120, 80, 130, 155, 135, 156, 213, 180, 90, 120, 75, 90, 90, 154, 142, 60, 28,
    96, 122, 174, 120, 208, 96, 120, 180, 120, 155, 150, 195, 140, 96, 96, 148, 36, 180,
    96, 120, 80, 103, 130, 132, 120, 90, 140, 75, 87, 111, 138, 140, 117, 120, 61, 75, 150,
    113, 100, 120, 150, 96, 136, 90, 90, 120, 120, 96, 180, 122, 144, 70, 160, 100, 180,
    116, 180, 120, 66, 150, 240, 60, 50, 165, 92, 165, 135, 60, 100, 125, 240, 100, 180,
    150, 100, 50, 92, 130, 180, 146, 184, 155, 155, 28, 85, 112, 110, 240, 180, 90, 133,
    117, 210, 125, 30, 28, 220, 150, 105, 150, 150, 165, 120, 112, 180, 60, 120, 205, 62};
```

```
In[ ]:= durationFill = {92, 60, 49.97363636, 167, 190, 97.90931131, 116.813375, 143, 130, 120, 180,
    135, 200, 64, 128, 90, 160, 112, 140, 120, 90, 120, 120, 120, 96, 75, 122, 250, 147.918255,
    160, 126.5531347, 180, 236, 180, 140, 108.5816943, 150, 80, 105, 135.8946952,
    65.93419376, 108, 108, 108, 108, 108, 108, 108, 108, 108, 149.5978733, 150,
    121.1241837, 150, 162, 120, 105, 252, 175, 92, 75, 120, 90, 121.1241837, 116.4483079,
    120, 105, 105, 96, 49, 163, 180, 94.22779415, 107.9277565, 78.18453952, 105, 210,
    123.165793, 130, 102, 111.7361663, 198, 75, 71.96203636, 85, 130, 96, 128, 108,
    146, 88, 144, 60, 117.3468317, 79.20597189, 125, 141.2879682, 38, 94.25467368,
    216, 90, 55, 148.8387732, 236, 160, 140, 160, 112, 101.2452893, 180, 137.0705455,
    92.21325758, 92, 120, 80, 120, 83.63706186, 180, 160, 169.4106273, 120, 121.1241837,
    80, 130, 155, 125.1038315, 115.51729, 135, 156, 213, 180, 90, 120, 99.92687532,
    100.6737065, 129.3118353, 75, 325, 210, 90, 90, 154, 29.17598285, 142, 60, 28, 96,
    122, 174, 120, 208, 111.4871731, 96, 120, 139.6523537, 180, 108.1016726, 120, 155,
    150, 195, 110.7789969, 140, 96, 96, 107.0863636, 148, 162.2520661, 121.1241837, 70,
    36, 113.1099716, 180, 96, 39.77493506, 39.77493506, 120, 80, 103, 130, 132, 120,
    90, 140, 75, 121.1241837, 87, 93.3378563, 111, 138, 140, 117, 120, 61, 75, 150, 113,
    100, 139.1718628, 120, 150, 74.13671329, 84.47924242, 96, 136, 90, 90, 55.30943633,
    120, 120, 96, 180, 122, 144, 70, 160, 100, 109.9854317, 180, 116, 180, 120, 66, 150,
    240, 60, 50, 102.8029091, 165, 92, 106.8823896, 165, 135, 60, 100, 128.5036364,
    125, 35.13235875, 240, 100, 180, 150, 19.84247326, 100, 50, 92, 65.93419376, 130,
    180, 146, 184, 155, 155, 28, 85, 178.4927444, 112, 114.8408934, 127.5844907,
    124.4274457, 110, 240, 180, 67.54678322, 90, 80, 133, 111.3698182, 117, 210,
    97.35123967, 125, 30, 65.93419376, 28, 220, 180.6839008, 318.7746873, 20.75827972,
    97.51998182, 150, 116.4158574, 19.27554545, 105, 150, 150, 165, 65.91696807, 120,
    129.2824463, 6.395435606, 112, 180, 60, 120, 79.7453867, 205, 101.7320455, 62};
```

```
In[ ]:= durFitData = Transpose[{lengthSpeed, durationSkips}];
```

Finished Data

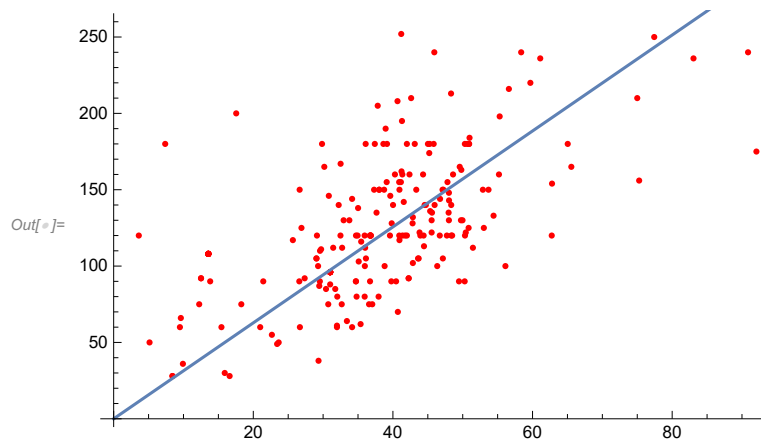
```
In[ ]:= durFit[x_] = Fit[durFitData, {x}, x]
```

```
Out[ ]:= 3.1412 x
```

```
In[ ]:= durFitDataPlot = ListPlot[data, PlotStyle -> Red];
```

```
In[ ]:= durFitPlot = Plot[durFit[x], {x, 0, 10000}];
```

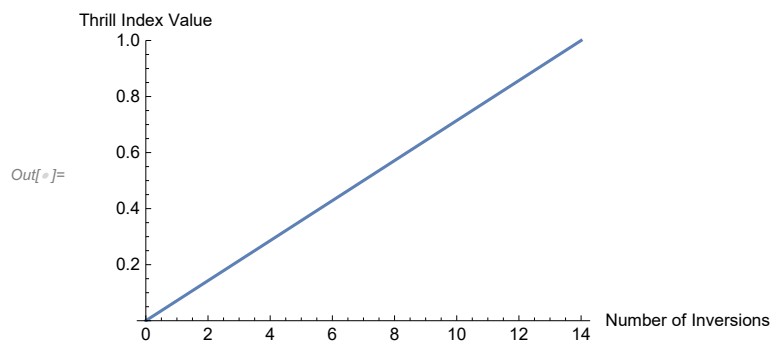
In[]:= Show[durFitDataPlot, durFitPlot]



Appendix C:

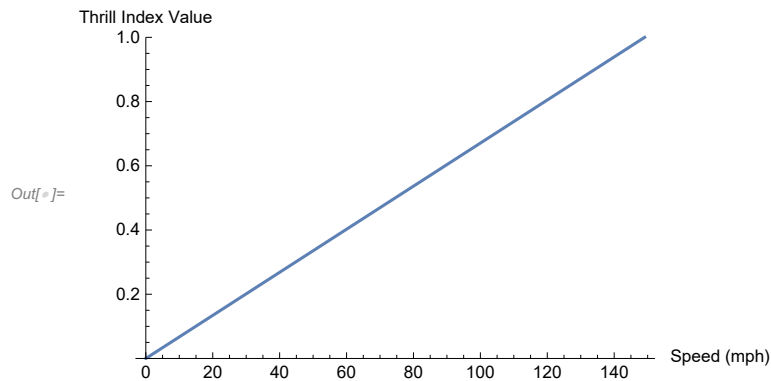
Inversions Function:

In[]:= InversionFunct[x] = $\frac{1}{14}x$;



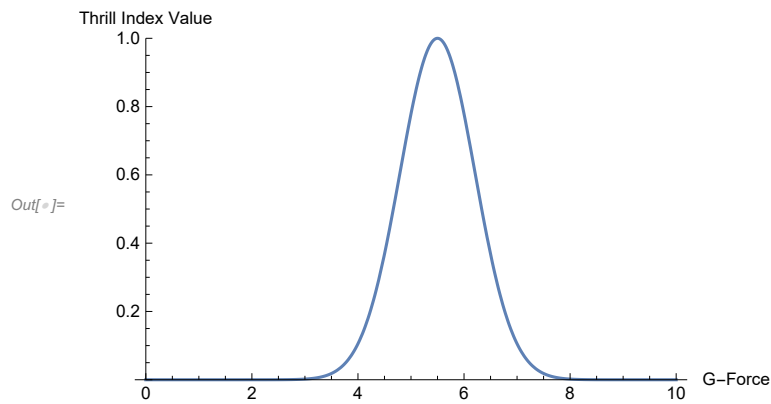
Speed Function:

In[]:= SpeedFunct[x] = $\frac{1}{149.1}x$;



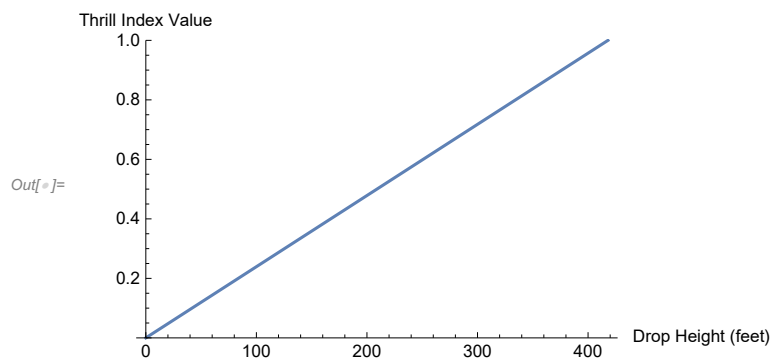
G-Force Function:

In[]:= **GForceFunc** [x] = $e^{-(x-5.5)^2}$;



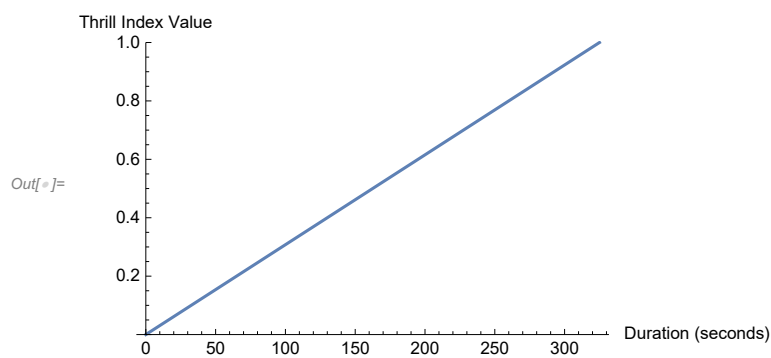
Drop Height Function:

In[]:= **DropHeightFunc** [x] = $\frac{1}{418} x$;



Duration Function:

In[]:= **DurationFunc** [x] = $\frac{1}{325} x$;



Raw excel equation:

$$=EXP(-1*POWER(R2-5.5,2))+(1/14)*O2+(1/149.1)*L2+(1/418)*P2 + (1/325)*T2$$

Where R2 = G-Value, O2 = Inversions, L2 = Speed, P2 = Drop Height, T2 =Duration

Normal (Real Life) equation: $e^{-(R2 - 5.5)^2 + 1/14 * O2 + 1/149.1 * L2 + 1/418 * P2 + 1/325 * T2}$.