Methodical Identification of High-Risk Space Debris

Marshall Vielmetti[1*]

[1*]Nicolet High School, Jean Nicolet, Milwaukee, 53217, Wisconsin, USA.

Corresponding author(s). E-mail(s): thomas.vielmetti@nicolet.us;

Abstract

Every passing year, more and more material is introduced into Earth's orbital environment. While regulations regarding end of life removal have steadily evolved over the past decades, hundreds of satellites exist with no means for safe removal. Concurrently, anti-satellite tests, orbital collisions, and the increasing commercialization of space further threaten the sanctity of the orbital environment. This study builds on the work of Donald Kessler and others who've worked to model the development of the orbital debris environment by using modern data, comparing the results to those of past studies, and introducing novel mathematical modeling approaches to identify debris objects that pose the greatest threat to the most orbital infrastructure in the event of a catastrophic collision event. To achieve this, a mathematical model was created that factored in 5 key attributes, including the spatial density of an orbit, risk of orbital instability, and length of debris life, to determine which specific debris objects pose the greatest threat. This model was able to rank every piece of debris by this weighted risk level. These rankings should be used in the future to inform decisions regarding debris removal as those technologies begin to come online. In future studies, additional information could be used to improve the accuracy and usefulness of the model's results. Information, specifically relating to the mass and size of specific debris objects, could be used to estimate the amount of debris objects created in a collision event, as well as improve the accuracy of orbital modeling.

Keywords: Space Debris, Mathematical Modeling, Kesslers Syndrome

1 Introduction

The occurrence of Kessler's syndrome has the possibility to devastate existing orbital infrastructure, such as communication and navigation satellites, as well as prevent the long-term buildup of space-related infrastructure that will be crucial as humanity progresses further into the space-age. Loosely, Kessler's syndrome can be understood as the cascading buildup of space debris– that once an orbit reaches a certain critical density, any collision would set off a chain-reaction that, over time, would cause the amount of debris in that orbit to increase faster than it would be removed naturally. In the long run, this would cause a significant decrease in the lifespan of orbital infrastructure, as ever-increasing debris clouds make collisions between large orbiting objects unavoidable rather than a once-in-a-decade occurrence. Not much has been done up to this point to prevent this from occurring. Modern satellites are required to implement a plan for end of life disposal, however this is not universally upheld, and existing infrastructure, especially that launched prior to the implementation of robust space-related legislation, will orbit until it either experiences a collision, is removed due to atmospheric drag, or a promising 3rd option– it is safely, and intentionally, de-orbited. Modern advances in technology promise to allow the intentional removal of satellites–without creating a large debris-cloud. The purpose of this study is to create a quantitative method to rank debris objects based on the threat they pose to surrounding orbital infrastructure, to inform the targeting of future debris-removal missions. This study draws upon concepts of Kessler's syndrome to analyze the long-term impacts to the orbital environment posed by tracked pieces of debris, and uses a system of mathematical and probabilistic models to assign each piece a "score", which determines their priority for removal. The study also makes a distinction between functional/maneuverable satellites, defunct satellites, and orbital debris. This allows the model to factor in the likelihood of collision independent of satellite-maneuverability, which is important in creating a robust, reliable model. This analysis uses original code written in Julia to perform all visualizations and modeling.

2 Previous Studies

2.1 Origins of Kessler's Syndrome

The concept known as Kessler's syndrome as first introduced in a 1978 paper titled "Collision frequency of artificial satellites: The creation of a debris belt". [1] In this highly influential paper, Kessler posited that satellite collisions in orbit, and the subsequent debris they would create, would cause a cascading increase in the occurrence of additional collisions. This would in turn create an orbital belt of debris surrounding the earth, potentially creating altitude bands at which placing satellites would be impossible. To identify the risk of this occurring, Kessler used a system of mathematical models to predict the rate at which such a debris belt could form. Using information from NORAD, Kessler calculated values for spatial density of orbital objects at different altitude bands.

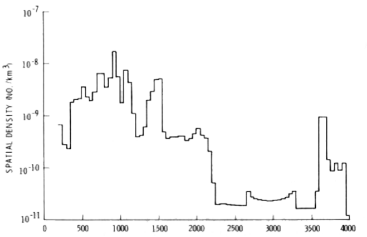

Fig. 1: Figure showing spatial density of orbiting objects by height in 1978. [1]

Kessler then introduced (1) to calculate the probability a given object experiences a collision.

$$dI/dt = S\overline{V}_s A_c \tag{1}$$

where $\overline{V}_s$ is the average relative velocities, A_c is the cross-section area of the object, I is the total number of impacts at time t, and S is the spacial density of

the altitude band. This equation can then be integrated as shown in (2) to determine dC/dt, the collision rate between all satellites.

$$dC/dt = \frac{1}{2} \int S^2 \overline{V}_s \overline{A}_{cc} dU \qquad (2)$$

where C is the total number of collisions between satellites, $\overline{A}_{cc}$ is the average cross-sectional area, and dU represents a small volume. Kessler uses these equations to show that the altitude band between 800km and 1000km was unstable, and at risk of undergoing cascading collisions. In a 1991 follow-up study [2], Kessler further defined what factors contributed to a band reaching critical density. Additional modeling, as well as more accurate data, further confirmed that the region between 800km and 1000km was unstable, and predicted that the region between 1000km and 1400km was approaching instability.

Kessler's findings were again corroborated in 2008 by [3], which found that even with no further satellite launches after 2005, the rate of debris growth due to collision would outpace the rate of removal. And assuming launches continue to grow at the expected exponential rate, this study proposes that the actual impacts will be even more significant than they predicted. The study used Monte-Carlo simulations to propagate the 2008 orbital environment 200 years into the future and analyze the rate of collision and debris growth.

2.2 Understanding Significance

There are currently 42,000 orbital debris objects with diameters greater than 10cm being tracked. [4] This is a significant increase from the 4,000 identified objects Kessler originally modeled in 1978 [1], and this figure has continued to grow at near exponential rates.

Another study by market research firm Visiongain estimates that the orbital environment is going to continue to become

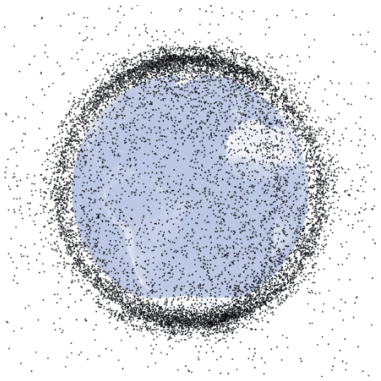

Fig. 2: Rendered image of space debris. Credit: NASA Orbital Debris Program

exponentially further crowded as the commercial space launch services market is expected to grow from US $ 9 billion to nearly US $ 50 billion by 2030, a CAGR of nearly 15%. This is driven primarily by the growth of various planned mega-constellations, such as SpaceX's StarLink, planned to consist of over 15,000 satellites. [5].

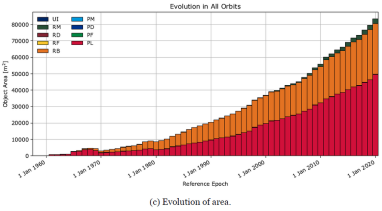

Fig. 3: Evolution of the total estimated cross-sectional area of space debris over time. Credit: European Space Agency [6]

Space debris poses immense risk to other objects in orbit. Because objects are often moving in speeds excess of 15,000mph [7], even the smallest of objects are a significant danger to any piece of orbital infrastructure. Using statistical analysis of returning spacecraft, NASA estimates that there could be half a million pieces of debris 1cm or larger, and as high as 100 million pieces 1mm or larger. These pieces of nearly untraceable

debris have been shown to pose the greatest mission-ending risk to LEO spacecraft. [7]

2.3 Efforts to Limit Debris Growth

In light of the predictions put forward by Kessler and others, global space agencies have been implementing rules and regulations to limit the growth of orbital debris. The first regulations, the Orbital Debris Mitigation Standard Practices (ODMPS), were put in place in 2001, and were recently overhauled in 2019 to account for modern improvements in spaceflight technology and mounting concerns over space debris. [8]. The goal of these and other regulations are to limit the growth of long-term debris in the orbital environment, and establish requirements about post-operational removal. Included in the ODMSP are guidelines for how much debris missions can reasonably allow to enter LEO, and the requirement that the probability of mission collision with an object larger than 10cm is less than 1 in 1000 over a 100 year operational life-cycle. The requirements also establish possibilities for post-mission disposal. Satellites can be sent into earth-escape orbits or directly re-entered into Earth's atmosphere, or otherwise put in what's commonly referred to as a graveyard orbit, where a satellite can be left where it poses little to no risk to functioning satellite constellations. Governments and companies around the world are also examining ways to remove existing debris from the atmosphere. The European Space Agency (ESA) has recently commissioned the first dedicated debris-removal program, set to launch in 2025. The mission is going to target the Vega Secondary Payload Adapter (VESPA) that was left in orbit as a result of the second flight of the ESA's Vega launcher in 2013. [9]. In the future, missions such as this, which is set to launch in 2025, could play a vital role in cleaning the orbital environment. Due to the large volume of satellites launched before strict government regulations existed around post-operational disposal, or even requirements about maneuverability to avoid debris, many objects such as VESPA exist in orbits without the ability to leave, or avoid other objects, and thus pose a grave danger, especially to the already overcrowded LEO environment. De-orbiting these dangerous objects will prove crucial to stabilizing the debris environment [10].

3 Research Plan

3.1 Goals

Overall Goal: Create a mathematical model that can be used to rank and identify pieces of debris for removal based on the threat they pose to the orbital environment.

1. **Subgoal:** Develop visuals to analyze the current state of the orbital environment.
2. **Subgoal:** Use visuals and analysis to target model and inform results.
3. **Subgoal:** Develop model based on Kessler's work and original analysis
4. **Subgoal:** Use model to generate list of highest risk objects for targeting by future removal programs.

3.2 Expected Outcomes

The expected outcome of this project is to output a list of 100 debris objects which pose the greatest threat, and to be able to support this conclusion with mathematical evidence.

3.3 Tools

1. Julia (High Performance, High Level Programming Language)
2. Julia's SatelliteToolbox Package - Very useful for accessing and interfacing with TLEs and orbital items
3. Desktop environment for programming and running analysis

3.4 Procedure

Because this is primarily an engineering project, with the goal of creating an innovative and unique solution, a highly detailed procedure is both unnecessary and would also serve to limit creativity

and innovation. Instead, the researcher chose to establish a broad structure of requirements, and work out the specifics in the moment.

1. Gather and clean (if necessary) satellite TLE (Two Line Element) data
2. Perform exploratory analysis of the orbital environment & create helper functions that will be used later.
3. Explore Kessler's original work and introduce some follow-on / updated to 2021 conclusions.
4. Begin more in-depth and final analysis. Explore spatial density and the distribution of objects, specifically looking across altitude bands.
5. Narrow focus & begin identifying model requirements and specifically creating targeted models to help with building this model.
6. Identify & build necessary mathematical equations to permit modeling approach.
7. Build model in Julia
8. Analyze and iterate based upon model results
9. Output model results to spreadsheet and draw final conclusions.

3.5 Risks

No risks were identified for this project.

3.6 Data Analysis

This entire project is built upon constantly elevated levels of data analysis. The final conclusions of the model will be evaluated by comparing the identified satellites to the distributions found in earlier sections of work & evaluating the results. Because no quantitative method exists, as it is the purpose of this paper to develop such a model, it is difficult to create a way to validate the model outside of critically evaluating the model's output. All of the data analysis will be done in Julia, using the default Plots package, as well as other standard practice analysis tools.

4 Orbital Environment Analysis

Using data obtained from [4], it is possible to replicate aspects of Kessler's original study. Determining the altitude of the elements yields the distribution found in Figure 4

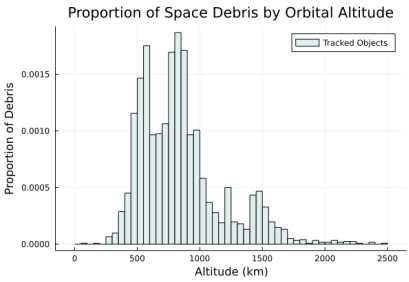

Fig. 4: Proportion of orbital objects found at each altitude.

The chart clearly shows peaks around the 500km and 1000km altitudes, and another, much smaller peak around 1500km. Not pictured, nor included in the proportion calculations, is a small peak around $35,000$km, where geostationary satellites orbit.

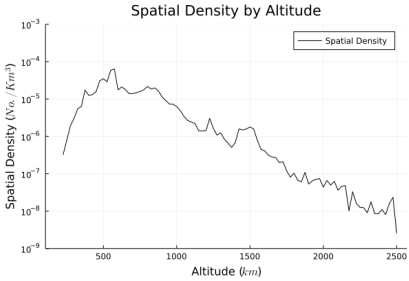

Fig. 5: Spatial density of objects by altitude.

Examining current spatial density is alarming. As shown by Figure 5, the area around 600-700km has very large spatial density values. In fact, the entire altitude

band between 400km and 900km has a spatial density greater than 10^{-6} objects per km^3. The peaks originally identified in Figure 1 were around 10^{-8}.

Equation 2 allows us to estimate collision frequency. Using Kessler's values of $\overline{V}_s \approx 7km/s$ and $\overline{A}_{cc} \approx 4m^2$, which were found to be approximately constant for altitudes under $2000km$. Integrating over each altitude band yields a collision rate of approximately 0.497 / year. Given that the number of catalogued objects has increased more than 10 times since Kessler's original study, this is a reasonable figure. It is also inline with what Kessler observed in 2009, when re-running his same analysis, wherein he found an estimated 0.13 collisions per year, with only around 13,000 tracked objects.

Additionally, binning by orbital inclinations allows a more fine-grained view of the situation, as shown in 6. Clearly visible are the highly populated orbital altitudes and inclinations. For instance, bands around 100, 80, 75, and 55 degrees of inclination are clearly present.

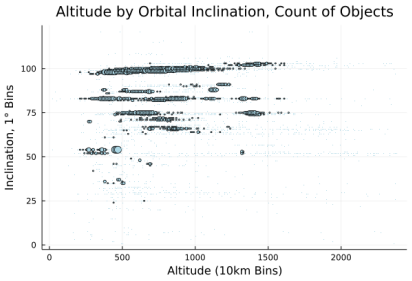

Fig. 6: Count of tracked objects by altitude and orbital inclination.

Furthermore, filtering by debris objects, and rerunning the previous analysis yields Figures 10, 8 9. While many of the previous bands are still visible, notably missing is the large concentration present around 50 degrees of inclination. Some digging reveals that the band is present due to SpaceX's starlink satellite constellation, which orbits at around 550km. This is intentional - at this altitude, objects at this altitude will experience enough atmospheric drag that without external power, they would succumb to orbital drag within a few years [11]. It should be noted that classified debris objects included in these calculations represent over half the the catalogued objects. However, not included in this list are objects of less than 10cm (which are difficult to track for current radar technology), as well as tracked satellites which may act as debris. Many satellites lack any ability to maneuver under their own power, and thus act as debris objects for the purpose of calculation collision probability. Due to TLE naming conventions, however, and a lack of a clear database or availability of this information, they are not included in these calculations.

The spike of debris (relative to surrounding orbits) shown by Figure 10 at 480km is likely the result of a Russian anti-satellite test conducted in November of 2021, and generated an estimated 300-400 pieces of 10cm or larger debris [12]. This debris, because it is in such a low orbit, will likely decay within 5-10 years. The impact of a similar anti-satellite test conducted by China in 2007, however, will be much more long lasting. In this test, the Chinese military destroyed a defunct weather satellite orbiting around 870km. This test is responsible for the large peak around that altitude, and due to higher orbit, this debris could last for a much longer period. This impact is estimated to have created over 1000 debris objects greater than 10cm in diameter, and the debris cloud extends throughout most of LEO. [13]. These two tests alone illustrate the risk that further testing poses to the orbital environment.

Also present are the results of a 2009 collision between between two Russian Satellites orbiting around 800km, which is also estimated to have created over 1000 debris objects of size 10cm or greater. [14]

There are many other orbital fragmentation / collision occurrences of note, and they are occurring with increasing frequency as the orbital environment becomes more populated. [15]

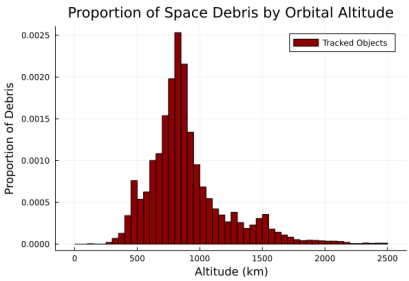

Fig. 7: Proportion of orbital debris objects found at each altitude.

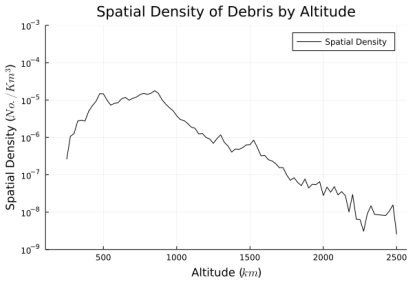

Fig. 8: Spatial density of debris objects by altitude.

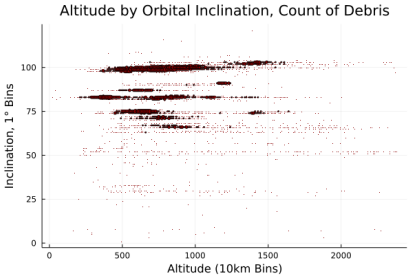

Fig. 9: Count of tracked orbital debris objects by altitude and orbital inclination.

These figures on their own, while insightful in their own right, are difficult to draw concrete conclusions from.

Applying Equation 2 to Figure 6 shows a very fine grained view of the most at risk orbits.

5 Comparing Debris to Overall Object Concentrations

To better understand the risk posed by debris to the orbital environment, it is important to understand how much debris exists in a given orbit. Using this information, it is possible to begin developing a mathematical model to assign "risk" posed by different debris concentrations, and ultimately specific objects, to their respective orbits.

A rudimentary analysis (Figure 10) shows that the proportions of debris to the total objects are highly variable when analyzed by altitude.

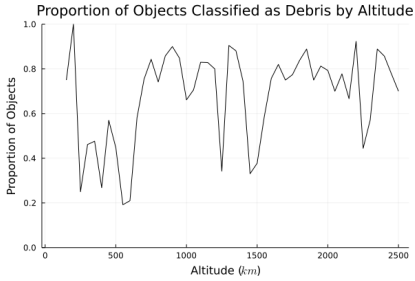

Fig. 10: Proportion of objects classified as debris by altitude.

This concept, however, serves as the baseline for more advanced analysis. In order to create a metric to analyze the risk posed by a piece of debris, 5 key metrics will be considered:

1. The number of non-debris objects in and around the orbit.
2. The overall spatial density of the orbit.
3. The number of debris objects in and around the orbit (to calculate probability of a collision occurring).
4. The altitude of the orbit and the estimated time for newly created objects to decay.
5. The impact a collision would have on the stability of the orbit as a whole.

Items 1, 2, and 3 have been adequately shown. Item 4 can be quickly estimated

using information from [16], which dictates that on average, orbits at heights less than 600km decay in a matter of years. Closer to 800km, the length to decay is a number of decades, and above 1000km, debris could last centuries before eventually falling to Earth. While a more precise method to calculate lifespan could be used, this information is accurate enough for the purposes of this study. Therefore, the only remaining information needed is item 5. This metric will use the broader concepts of Kessler's syndrome to estimate a spatial density at which more debris would be created by collisions than removed naturally through decay.

6 Modeling Environment Evolution

There are two keys components of this portion of the model. One metric needs to be established to determine whether or not an orbit is "unstable"– that the estimated rate of debris growth due to collision would outpace removal. This model also has to calculate the probability a specific piece of debris collides with either another piece of debris or a satellite object and the resulting debris that would occur, which could then be fed in to the earlier part of this model to determine whether or not such a collision would cause instability. This would output whether or not the piece of debris poses a risk to destabilize in the event of a catastrophic collision event, and how likely such a collision is to occur. The likelihood of a specific object experiencing a collision is given to by equation 1. Because data regarding the mass of given objects is not readily available, it is difficult to give an accurate figure with regards to the number of fragments produced as a result. In the case of a collision between two satellites, we know that historically between 400 and 2300 objects of size greater than 10cm have been produced. Because this model would be looking primarily at the debris produced by collisions between a small debris object and a satellite, it is reasonable to nearly halve this number, and assume that in such a collision the mass of the debris object to be negligible. Therefore, it is reasonable to assume that on the low end, approximate 400 objects would be created as the result of a collision between a debris object of negligible mass and a satellite. Is is also important to note that because of the extreme velocities that orbiting objects travel, an object approximately the size of a golf ball could cause the complete fragmentation of most satellites [7]. Using this logic, it is also reasonable to assume that the collision of two pieces of debris to produce a negligible amount of debris objects. Therefore, in the development of this model, the primary focus will be on the probability of collision between a debris object of negligible mass and a intact, defunct satellite object, or a collision between two defunct satellites. For the purpose of this analysis, the ability of a satellite to maneuver will be associated directly with whether or not the satellite is cur es. According to the European Space Agency, of the 7,840 satellites currently in space, only 5100 are currently functioning, therefore approximately 35% of non-debris, satellite objects lack the ability to maneuver. This distribution will be assumed constant across all altitude / inclination bands. With these figures in mind, analysis across altitude yields Figure 11. Note that the count of "Non-Functional Satellites" is simply 35% of the "Functional Satellites".

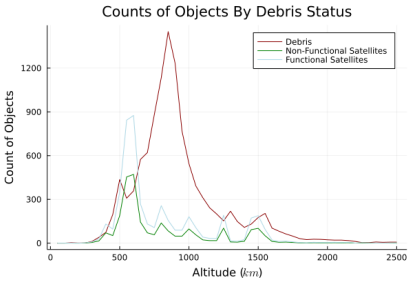

Fig. 11: Count of objects by status as debris by altitude.

Because functional satellites are deemed to posses the ability to maneuver for the purposes of this study, their

probability of collision at current debris levels is assumed to be 0. This is assumed to become false as debris levels increase. This is also inline with common sense observations; while the occasional maneuver to dodge debris is practical, having to be constantly changing orbits to avoid an ever growing debris cloud quickly becomes unrealistic.

Integrating equation 2 over these altitude bands yields more interesting results. Figure 12 illustrates the high collision frequency, especially peaking at around 550km altitude, which then steadily drops off with another slight peak around both 1300km and 1500km before rapidly decreasing.

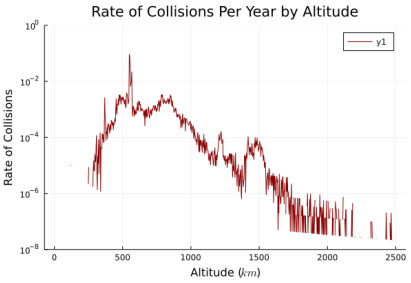

Fig. 12: Rate of collisions per year by altitude. This is very similar, and strongly related, to the graph of spatial density.

To further develop the model, it is now necessary to examine the rate of collisions between specific subsets of objects. Figure 13 shows the rates of collisions between pairs of non-functional satellite objects by altitude, and figure 14 shows the rates of collisions between non-functional satellite objects and debris, by altitude. Together, the model predicts an average of 0.147 collisions / year, or approximately 1 every 7 years, which is consistent with the rate of satellite-related catastrophic breakup events we have been observing for the last decade. This approach is more accurate for determining future debris concentrations than simply running the analysis for every satellite object, as many collisions, such as those between small debris objects, will generate much smaller amounts of debris than satellite-satellite collisions, and this also accounts for the ability of many modern, functioning satellites to avoid most space debris.

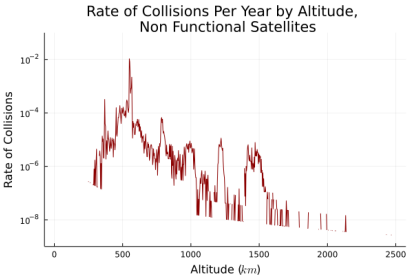

Fig. 13: Rate of collisions per year by altitude, among only non-functional satellites. The sum of these collision rates yields a rate of 0.033 collisions / year, or one such collision every 30 years.

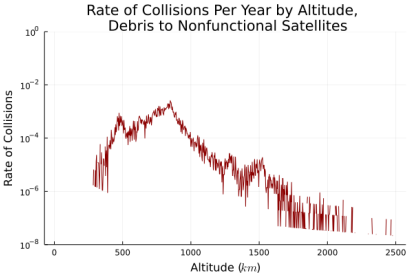

Fig. 14: Rate of collisions per year by altitude, between non-functional satellites and debris objects. The sum of these collision rates yields a rate of 0.114 collisions / year, or one such collision approximately every 9 years.

Also useful is equation 3, which was derived by Kessler in [2], and is used to calculate the critical number of objects which would produce a runaway orbital environment at a given altitude.

1. ${}_R N_i$ is the number of intact objects above h1 which results in a runaway environment
2. $a = r_e + h_i$, where r_e is the radius of the earth, and h_i is the altitude of interest
3. V_o is the orbital velocity

4. ρ_a is the atmospheric density
5. C_D is the atmospheric drag co-efficient (which Kessler found to be approximately equal to 2.2)
6. N_o is the number of objects with enough mass to break up a satellite
7. W is a measure of orbital eccentricity (1 for circular orbits), which can be approximated as 1.5 for times after a collision.
8. $(m/A)_a$ is the average mass over area - value of 125 is used, as derived by Kessler, and assumed to be constant across the entire distribution
9. V is the average relative velocity, and equals approximately $7.5km/sec$
10. σ_f is the collision cross-section between an intact object and a fragment, assumed to be 14 m^2, as derived by Kessler.

$$_R N_i = \frac{4\pi a^3 V_o \rho_a C_D}{N_0 W (m/A)_a V \sigma_f} \tag{3}$$

Evaluating this equation yields figure 15 when compared with current satellite debris counts

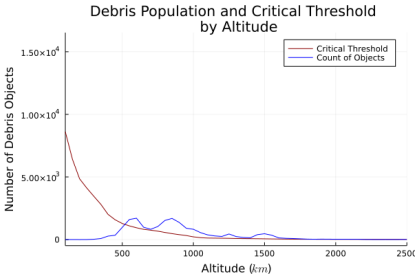

Fig. 15: Critical debris threshold (count of debris objects) as dictated by 3, and count of objects by altitude.

A number of things stand out about this graph. Primarily, the observed trend appears to be inverted from what is expected. This is, however, due to the role of atmospheric drag. Even though the actual rate of collision would be much lower, because of the much longer time spans at which debris exists at high altitudes, even small amounts will ultimately result in cascading. At much lower altitudes, however, due to the very high atmospheric drag, debris rapidly returns to Earth, so much more is required for the rate of growth due to collision to outpace this rate of removal. According to this data, much of our current orbital environment, especially that between 550 and 1500, is unstable, or at risk of run-away collisions. This is in line with what has been observed by Kessler and others.

7 Final Model Development

With this, it is possible to begin the implementation of the final model.

Immediately, based on figure 15, this model will discount any objects orbiting below 500km. Their time in orbit is too short to be practically considered, and the required debris population to have a large impact on the long-term outlook of the orbit is too large to feasibly occur.

It also makes sense to limit the search to defunct satellites and other structural pieces of debris. These are more likely to have a high mass than a given debris object, and thus, as previously explained, are likely to produce more fragments in the event of collision.

From equation 1 and figures 13 and 14, it is possible to estimate the rate at which a defunct satellite object will experience collisions. S_d represents the spatial density of debris objects in the altitude band, and S_s the spatial density of defunct satellites in the altitude band. P_s represents the probability that the object experiences a collision over the course of a year.

$$P_s = (S_d + S_s)\overline{V}_s A_c \tag{4}$$

Once the likelihood of an object experiencing a collision is quantified, the estimated debris produced by an impact must be analyzed. Let D_i represent this estimated debris produced in 5. $\overline{X}_d$ and $\overline{X}_s$ represent the average number of fragments created in a debris - satellite and satellite - satellite collision, respectively. σ_d and σ_s represent the proportions of objects in the altitude band which are debris and satellites, respectively. Thus, $\sigma_s + \sigma_d = 1$

$$D_i = \overline{X}_d \sigma_d + \overline{X}_s \sigma_s \tag{5}$$

For the purposes of this study, the values $\overline{X}_d = 400$ and $\overline{X}_s = 1000$ will be used– as shown previously, these arise naturally from observed counts of debris produced by prior collisions. The exact figures here are also not of large importance to the reliability of this information, as they merely act as scaling factors in either direction. In a future study, with access to accurate mass information for tracked objects, these could be used to significantly improve the accuracy of the model's results.

Together, equations 4 and 5 lead to 6, where D_p is the average amount of debris produced over the course of a year.

$$D_p = D_i P_s \tag{6}$$

To begin to quantify the impact of this debris, equation 7 is introduced. ΔP_f represents the average change in collision probability experienced by functioning satellites in the same altitude band as the target object as a result of the presence of the specified piece of orbital debris, and as a function of its probability of undergoing collision. $\Delta S = S_1 - S_0$, where S_1 represents the spatial density adjusted for D_p, and S_0 represents the baseline levels. The other variables are the same as in equation 2. This integration should only be carried out over the altitude band that contains the target satellite.

$$\Delta P_f = \frac{1}{2} \int \Delta S^2 \overline{V}_s \overline{A}_{cc} dU \tag{7}$$

An object with a large ΔP_f value would, on average, be more likely to cause damage to a large number of functioning satellites than another object. In addition to this figure, however, also of importance is the impact of an objects' collision on the overall spatial density of its respective orbit. Specifically, it would be beneficial to quantify this in relation to the orbits stability. This can be done by evaluating the change as a result of the object experiencing a collision (and the likelihood of such a collision occurring) as a percentage of the overall requirement for instability. Letting this metric be α, equation 8 arises.

$$\alpha = \frac{D_p}{{}_R N_i} \tag{8}$$

With this, two factors have been created that reflect the 'threat' posed by a specific piece of debris; ΔP_f and α. Together, these reflect the average threat to orbital infrastructure posed by an object (ΔP_f), as well as the threat to overall orbit stability (α).

To determine the overall priority for an object, the model ranked every item relative to the others based on the 2 criteria, and added up their scores. For example, if an object had the largest ΔP_f and α, it would receive a score of 1 and 1 in each area, which would add up to 2. The objects with the lowest cumulative scores would be given the highest priority for removal.

8 Identifying Key Targets for Removal and Model Conclusions

The model ranked over 7,000 satellites / large orbital objects. After some manual filtering by the researcher, the model suggests that the highest priority satellites to remove are Magion 2, an autonomous research satellite launched nearly 40 years ago that orbits at around 1500km, followed by BLOCK DM-SL R/B, a rocket body left in orbit as a result of 2012 launch that orbits on a highly eccentric orbit, but has a perigee around 1400km. The third is the CZ-2C R/B rocket body, that orbits around 800km, and was the result of a 2009 Chinese rocket launch. This study has successfully created a quantitative method which can be used to analyze the potential impact of orbital debris. With the first test of active debris removal set for 2025. This model, and its furthered evolution, should be used to make sure to maximize the impact of every removal mission. The model functioned as intended, and successfully identified targets such as Magion 2, BLOCK DM-SL R/B, and CZ-2C R/B, which orbit in high-risk orbits, and pose a significant and real threat to existing infrastructure. While there is still

work to be done before this model is used to inform these missions, this study provides a powerful starting point for future work as well as a clear path forwards.

9 Next Steps

In order to improve the predictive ability of this model, further work should be done to incorporate additional data, such as cross section and estimated mass of specific objects. This would allow the model to better calculate the impact rate and resulting debris cloud of specific objects. Furthermore, additional work should be done to run the model with regards to more fine-tuned data, such as orbital inclination and eccentricity. This would allow more accurate analysis of the true spatial densities experienced by objects over time. This work could be further supplemented by incorporate simulated debris environment evolution into the study and modeling process. The goal of the researchers is to begin this process in the near future, and steps have already been taken regarding the data collection and cleaning necessary for such an undertaking.

Appendix A

All of the code used in the analysis models:

```
using SatelliteToolbox
import Random
using Plots
using StatsBase
using LaTeXStrings
gr()

global eop = get_iers_eop()

# The DCM (Direction Cosine Matrix) that rotates
# TEME into alignment with ITRF
global D_ITRF_TEME = rECItoECEF(TEME(), ITRF(),
    DatetoJD(2022, 1, 1, 0, 0, 0), eop)

function runAnalysis()
    tles = read_tle("TLEData.txt")

    altitudes = getTleAltitude.(tles)
    filter!((x) -> x .< 2500 && x .> 100, altitudes)

    return altitudes
end

function getTleAltitude(tle)
    orbp = init_orbit_propagator(Val(:sgp4), tle)
    try
        r_teme, v_teme = propagate_to_epoch!(orbp,
            DatetoJD(2022, 1, 12, 0, 0, 0))
        r_itrf = D_ITRF_TEME * r_teme
        lat, lon, h = ecef_to_geodetic(r_itrf)

        return h / 1000
    catch
        return 0
    end
end

# Returns tupple altitude, inclination
function getTleAltitudeInclination(tle)
    orbp = init_orbit_propagator(Val(:sgp4), tle)
    try
        r_teme, v_teme = propagate_to_epoch!(orbp,
            DatetoJD(2022, 1, 12, 0, 0, 0))
        r_itrf = D_ITRF_TEME * r_teme
        lat, lon, h = ecef_to_geodetic(r_itrf)

        return [h / 1000; tle.i]
    catch
        return [0; 0]
    end
end
```

```
function doVisualization(altitudes)
    histogram(altitudes, bins =
        range(0, step = 50, stop = 2500), label = "Tracked Objects",
        title = "Proportion of Space Debris by Orbital Altitude",
        color = "azure2", normalize = true)
    xlabel!("Altitude (km)")
    ylabel!("Proportion of Debris")
end

function doSpatialDensityCalculations(altitudes)
    bins = 150:25:2500

    h = fit(Histogram, altitudes, bins)

    bins = collect(bins)
    popat!(bins, 1)

    densities = getSphereDensity.(bins, h.weights)

    plot(bins, densities, color = "black", label = "Spatial Density",
        yscale = :log10, ylims = (10E-10, 10E-4))
    title!("Spatial Density by Altitude")
    xlabel!(L"Altitude ($km$)")
    ylabel!(L"Spatial Density ($No. / Km^3$)")
end

function getSphereDensity(radius, count)
    outerVolume = 4 / 3 * pi * radius^3
    innerVolume = 4 / 3 * pi * (radius - 0.005)^3

    sliceVolume = outerVolume - innerVolume

    return count / sliceVolume
end

#Main driver function for inclination analysis
function doInclinationAnalysis()
    # Plot relative densities in altitude by inclination - 2D histogram
    # Using 1km and 1 degree bins
    tles = read_tle("TLEData.txt")

    # m = matrix [altitude, inclination]
    m = getTleAltitudeInclination.(tles)

    m = mapreduce(permutedims, vcat, m)

    m = m[(m[:, 1].<2500).&(m[:, 1].>100), :]

    altitudes = m[:, 1]
    inclinations = m[:, 2]

    # weights array altitudes by inc
    d = fit(Histogram, (altitudes, inclinations), ((100:10:2500,
```

```
        0:1:130)), closed = :right)

    bindata = zeros(1, 3)

    for alt in 1:size(d.weights)[1]
        for incl in 1:size(d.weights)[2]
            if d.weights[alt, incl] > 0
                bindata = [bindata; [alt * 10 incl d.weights[alt, incl]]]
            end
        end
    end
    @show size(bindata)

    bindata = bindata[2:end, :]

    bindata[:, 3] = log.(bindata[:, 3])

    @show size(bindata)

    plot(bindata[:, 1], bindata[:, 2], seriestype = :scatter,
        markersize = bindata[:, 3], color = "lightblue", legend = false)
    title!("Altitude by Orbital Inclination, Count of Objects")
    xlabel!("Altitude (10km Bins)")
    ylabel!("Inclination, 1   Bins")
    png("Figures/PNG/Altitude by Orbital Inclination, Count of Objects")
    savefig("Figures/SVG/Altitude by Orbital Inclination,
        Count of Objects.svg")
    # histogram2d(m[:, 1], m[:, 2], bins = (100:10:2500, 0:1:130))
    # return altitudes
end

#Main driver function for inclination analysis for only deb objects
function doDebrisOnlyInclinationAnalysis()
    # Plot relative densities in altitude by inclination - 2D histogram
    # Using 1km and 1 degree bins
    tles = read_tle("TLEData.txt")

    # @show fieldnames(typeof(tles[1]))
    # @show tles[1].name
    # @show occursin("Deb", tles[1].name)

    filter!(e -> occursin("DEB", e.name), tles)

    # m = matrix [altitude, inclination]
    m = getTleAltitudeInclination.(tles)

    m = mapreduce(permutedims, vcat, m)

    m = m[(m[:, 1].<2500).&(m[:, 1].>100), :]

    altitudes = m[:, 1]
    inclinations = m[:, 2]

    # weights array altitudes by inc
```

```
    d = fit(Histogram, (altitudes, inclinations), ((100:10:2500,
        0:1:130)), closed = :right)

    bindata = zeros(1, 3)

    for alt in 1:size(d.weights)[1]
        for incl in 1:size(d.weights)[2]
            if d.weights[alt, incl] > 0
                bindata = [bindata; [alt * 10 incl d.weights[alt, incl]]]
            end
        end
    end
    @show size(bindata)

    bindata = bindata[2:end, :]

    bindata[:, 3] = log.(bindata[:, 3])

    @show size(bindata)

    plot(bindata[:, 1], bindata[:, 2], seriestype = :scatter,
        markersize = bindata[:, 3], color = :darkred, legend = false)
    title!("Altitude by Orbital Inclination, Count of Debris")

    xlabel!("Altitude (10km Bins)")
    ylabel!("Inclination, 1   Bins")
    png("Figures/PNG/Debris - Orbit Inclination x Altitude Graph")
    savefig("Figures/SVG/Debris - Orbit Inclination x Altitude Graph.svg")
    # return altitudes
end

function doDebrisAltitudeBinAnalysis()
    tles = read_tle("TLEData.txt")

    filter!(e -> occursin("DEB", e.name), tles)

    altitudes = getTleAltitude.(tles)
    filter!((x) -> x .< 2500 && x .> 100, altitudes)

    histogram(altitudes, bins = range(0, step = 50, stop = 2500),
        label = "Tracked Objects",
        title = "Proportion of Space Debris by Orbital Altitude",
        color = "darkred", normalize = true)
    xlabel!("Altitude (km)")
    ylabel!("Proportion of Debris")

    png("Figures/PNG/Debris - Proportion by Altitude")
    savefig("Figures/SVG/Debris - Proportion by Altitude.svg")

end

function doDebrisSpacialDensityAnalysis()
    tles = read_tle("TLEData.txt")
```

```
    filter!(e -> occursin("DEB", e.name), tles)

    altitudes = getTleAltitude.(tles)
    filter!((x) -> x .< 2500 && x .> 100, altitudes)

    bins = 150:25:2500

    h = fit(Histogram, altitudes, bins)

    bins = collect(bins)
    popat!(bins, 1)

    densities = getSphereDensity.(bins, h.weights)

    plot(bins, densities, color = "black", label = "Spatial Density",
        yscale = :log10, ylims = (10E-10, 10E-4))
    title!("Spatial Density of Debris by Altitude")
    xlabel!(L"Altitude ($km$)")
    ylabel!(L"Spatial Density ($No. / Km^3$)")

    png("Figures/PNG/Debris - Spatial Density")
    savefig("Figures/SVG/Debris - Spatial Density.svg")
end

function getStatistics()
    tles = read_tle("TLEData.txt")

    @show length(tles)

    filter!(e -> occursin("DEB", e.name), tles)

    @show length(tles)
end

function doProportionDebrisByAltitude()
    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris

    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)

    filter!((x) -> x .< 2500 && x .> 100, DEB_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, NOD_altitudes)

    bins = (0:50:2500)

    DEB_binned = fit(Histogram, DEB_altitudes, bins, closed = :right)
    NOD_binned = fit(Histogram, NOD_altitudes, bins, closed = :right)

    bins_arr = collect(bins)
    popat!(bins_arr, 1)
```

```
    proportions = DEB_binned.weights ./ (DEB_binned.weights .+
        NOD_binned.weights)
    plot(bins_arr, proportions, color = "black", legend = false,
        ylims = (0, 1))

    title!("Proportion of Objects Classified as Debris by Altitude")
    xlabel!(L"Altitude ($km$)")
    # xlims!(0, 1)
    ylabel!("Proportion of Objects")

    png("Figures/PNG/Comb - Proportion Debris Objects")
    savefig("Figures/SVG/Comb - Proportion Debris Objects.svg")

end

function doCountDebrisNonDebrisComparisonByAltitude()
    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris

    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)

    filter!((x) -> x .< 2500 && x .> 100, DEB_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, NOD_altitudes)

    bins = (0:50:2500)

    DEB_binned = fit(Histogram, DEB_altitudes, bins, closed = :right)
    NOD_binned = fit(Histogram, NOD_altitudes, bins, closed = :right)

    bins_arr = collect(bins)
    popat!(bins_arr, 1)

    DEB_ADJ = DEB_binned.weights
    NONF = NOD_binned.weights .* 0.35 #Non functional satellites
    NOD_ADJ = NOD_binned.weights .* (1 - 0.35) #Functional satellites

    VALS = [DEB_ADJ NONF NOD_ADJ]

    plot(bins_arr, VALS, color = ["darkred" "green" "lightblue"],
        legend = true, label = ["Debris" "Non-Functional Satellites"
        "Functional Satellites"])

    title!("Counts of Objects By Debris Status")
    xlabel!(L"Altitude ($km$)")
    # xlims!(0, 1)
    ylabel!("Count of Objects")

    png("Figures/PNG/Comb - Count of Objects By Debris Status")
    savefig("Figures/SVG/Comb - Count of Objects By Debris Status.svg")
```

```
end

function calculateBandedCollisionFrequency()
    # Calculate spatial density of all bands with all objects
    # Calculate collision frequency by band

    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris

    ALL_altitudes = getTleAltitude.(tles)
    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)

    filter!((x) -> x .< 2500 && x .> 100, ALL_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, DEB_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, NOD_altitudes)


    sliceWidth = 3
    bins = (0:sliceWidth:2500)

    ALL_binned = fit(Histogram, ALL_altitudes, bins, closed = :right)
    DEB_binned = fit(Histogram, DEB_altitudes, bins, closed = :right)
    NOD_binned = fit(Histogram, NOD_altitudes, bins, closed = :right)

    bins_arr = collect(bins)
    popat!(bins_arr, 1)

    ALL = ALL_binned.weights

    DEB_ADJ = DEB_binned.weights
    NONF = NOD_binned.weights .* 0.35 #Non functional satellites
    NOD_ADJ = NOD_binned.weights .* (1 - 0.35) #Functional satellites


    ALL_DENSITY = getSphereDensity.(bins_arr, ALL)
    DEB_ADJ_DENSITY = getSphereDensity.(bins_arr, DEB_ADJ)
    NONF_ADJ_DENSITY = getSphereDensity.(bins_arr, NONF)
    NOD_ADJ_DENSITY = getSphereDensity.(bins_arr, NOD_ADJ)

    Acc_DEB = 0.5 # Estimate
    Acc_SAT = 4 #Kessler
    Vs = 7 #Kessler

    # CF_ALL = ALL_DENSITY .^ 2 .* Acc_SAT .* Vs .*

    # @show sum(CF_ALL)

    CF_ALL = 0.5 .* ALL_DENSITY .^ 2 .* Acc_SAT .* Vs .* ((4 / 3 *   ) .*
        ((bins_arr .^ 3) .- (bins_arr .- sliceWidth) .^ 3))
```

```
    @show sum(CF_ALL)

    # @show sum(CF_ALL) / (4 / 3 *    * (2500^3))

    plot(bins_arr, CF_ALL, yscale = :log10, ylims = (10E-9, 10E-1),
        color = "darkred")

    title!("Rate of Collisions Per Year by Altitude")
    xlabel!(L"Altitude ($km$)")
    # xlims!(0, 1)
    ylabel!("Rate of Collisions")

    png("Figures/PNG/Comb - Rate of Collisions by Altitude")
    savefig("Figures/SVG/Comb - Rate of Collisions by Altitude.svg")
end

calculateBandedCollisionFrequency()
# Probability of Debris - Nonf collision, by altitude band
function calculateBandedCollisionFrequencyDebToNonf()
    # Calculate spatial density of all bands with all objects
    # Calculate collision frequency by band

    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris

    ALL_altitudes = getTleAltitude.(tles)
    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)

    filter!((x) -> x .< 2500 && x .> 100, ALL_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, DEB_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, NOD_altitudes)


    sliceWidth = 3
    bins = (0:sliceWidth:2500)

    ALL_binned = fit(Histogram, ALL_altitudes, bins, closed = :right)
    DEB_binned = fit(Histogram, DEB_altitudes, bins, closed = :right)
    NOD_binned = fit(Histogram, NOD_altitudes, bins, closed = :right)

    bins_arr = collect(bins)
    popat!(bins_arr, 1)

    ALL = ALL_binned.weights

    DEB_ADJ = DEB_binned.weights
    NONF = NOD_binned.weights .* 0.35 #Non functional satellites
    NOD_ADJ = NOD_binned.weights .* (1 - 0.35) #Functional satellites
```

```
    ALL_DENSITY = getSphereDensity.(bins_arr, ALL)
    DEB_ADJ_DENSITY = getSphereDensity.(bins_arr, DEB_ADJ)
    NONF_ADJ_DENSITY = getSphereDensity.(bins_arr, NONF)
    NOD_ADJ_DENSITY = getSphereDensity.(bins_arr, NOD_ADJ)

    Acc_DEB = 0.5 # Estimate
    Acc_SAT = 4 #Kessler
    Vs = 7 #Kessler

    # CF_ALL = ALL_DENSITY .^ 2 .* Acc_SAT .* Vs .*

    # @show sum(CF_ALL)

    CF_DEB = 0.5 .* DEB_ADJ_DENSITY .^ 2 .* Acc_SAT .* Vs .* ((4 / 3 *   )
        .* ((bins_arr .^ 3) .- (bins_arr .- sliceWidth) .^ 3))

    CF_DEB ./ DEB_ADJ .* NONF
    # Scale to be collisions between nonf and deb rather than between deb

    @show sum(CF_DEB)

    # @show sum(CF_ALL) / (4 / 3 *    * (2500^3))

    plot(bins_arr, CF_DEB, yscale = :log10, ylims = (10E-9, 10E-1),
        legend = false, color = "darkred")

    title!("Rate of Collisions Per Year by Altitude, \n
        Debris to Nonfunctional Satellites")
    xlabel!(L"Altitude ($km$)")
    # xlims!(0, 1)
    ylabel!("Rate of Collisions")

    png("Figures/PNG/DEBTONONF - Rate of Collisions by Altitude")
    savefig("Figures/SVG/DEBTONONF - Rate of Collisions by Altitude.svg")
end


calculateBandedCollisionFrequencyDebToNonf()

#Probability of Nonf - Nonf collision, by altitude band
function calculateBandedCollisionFrequencyNonftoNonf()
    # Calculate spatial density of all bands with all objects
    # Calculate collision frequency by band

    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris

    ALL_altitudes = getTleAltitude.(tles)
    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)
```

```
    filter!((x) -> x .< 2500 && x .> 100, ALL_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, DEB_altitudes)
    filter!((x) -> x .< 2500 && x .> 100, NOD_altitudes)


    sliceWidth = 3
    bins = (0:sliceWidth:2500)

    ALL_binned = fit(Histogram, ALL_altitudes, bins, closed = :right)
    DEB_binned = fit(Histogram, DEB_altitudes, bins, closed = :right)
    NOD_binned = fit(Histogram, NOD_altitudes, bins, closed = :right)

    bins_arr = collect(bins)
    popat!(bins_arr, 1)

    ALL = ALL_binned.weights

    DEB_ADJ = DEB_binned.weights
    NONF = NOD_binned.weights .* 0.35 #Non functional satellites
    NOD_ADJ = NOD_binned.weights .* (1 - 0.35) #Functional satellites


    ALL_DENSITY = getSphereDensity.(bins_arr, ALL)
    DEB_ADJ_DENSITY = getSphereDensity.(bins_arr, DEB_ADJ)
    NONF_ADJ_DENSITY = getSphereDensity.(bins_arr, NONF)
    NOD_ADJ_DENSITY = getSphereDensity.(bins_arr, NOD_ADJ)

    Acc_DEB = 0.5 # Estimate
    Acc_SAT = 4 #Kessler
    Vs = 7 #Kessler

    CF_NONF = 0.5 .* NONF_ADJ_DENSITY .^ 2 .* Acc_SAT .* Vs .*
        ((4 / 3 *   ) .* ((bins_arr .^ 3) .- (bins_arr .- sliceWidth)
        .^ 3))

    @show sum(CF_NONF)

    plot(bins_arr, CF_NONF, yscale = :log10, ylims = (10E-3, 10E2),
        legend = false, color = "darkred")

    title!("Rate of Collisions Per Year by Altitude, \n
        Non Functional Satellites")
    xlabel!(L"Altitude ($km$)")
    # xlims!(0, 1)
    ylabel!("Rate of Collisions")

    png("Figures/PNG/NONF - Rate of Collisions by Altitude")
    savefig("Figures/SVG/NONF - Rate of Collisions by Altitude.svg")
end

calculateBandedCollisionFrequencyNonftoNonf()

function doCalculateRunawayThreshold()
    tles = read_tle("TLEData.txt")
```

```
    ALL_altitudes = getTleAltitude.(tles)

    filter!((x) -> x .< 2500 && x .> 100, ALL_altitudes)

    bins = collect(0:50:2500)

    ALL_binned = fit(Histogram, ALL_altitudes, bins, closed = :right)

    CUMULATIVE = cumsum(reverse(ALL_binned.weights))

    popat!(bins, 1)


    Re = 6371 # radius of the earth, km
    G = 6.673 * 10^-11 # Gravitational constant
    Me = 5.972 * 10^24 # Mass of earth, Kg
    Cd = 2.2
    W = 1.5

    mA = 125 # Average mass over area - given by Kessler
    V = 7.5 # Avg. Relative veloctiy, km/sec, given by Kessler
     f  = 14

    # rNi_arr = (4 .*    .* (Re .+ bins) .^ 3 .* .   ((G + Me) ./
        (Re .+ bins)) .* expatmosphere.(bins .* 1000) .* Cd) ./
        (CUMULATIVE .* W .* mA .* V .*  f )
    rNi_arr = (4 .*    .* (Re .+ bins) .^ 3 .* .   ((G + Me) ./
        (Re .+ bins)) .* 10^-11 .* Cd) ./
        (CUMULATIVE .* W .* mA .* V .*  f )

    @show rNi_arr

    plot(bins, [rNi_arr ALL_binned.weights], labels = ["Critical
        Threshold" "Count of Objects"], color = ["darkred" "blue"])

    title!("Debris Population and Critical Threshold \n by Altitude")
    xlabel!(L"Altitude ($km$)")
    xlims!(100, 2500)
    ylabel!("Number of Debris Objects")

    png("Figures/PNG/COMB - Critical Threshold")
    savefig("Figures/SVG/COMB - Critical Threshold.svg")
end


doCalculateRunawayThreshold()

function doDebrisRankingModel()
    tles = read_tle("TLEData.txt")

    getTleAltitude.(tles)
end
```

```
function buildAllFigures()
    global eop = get_iers_eop()

    # The DCM (Direction Cosine Matrix) that rotates
    # TEME into alignment with ITRF
    global D_ITRF_TEME = rECItoECEF(TEME(), ITRF(),
        DatetoJD(2022, 1, 1, 0, 0, 0), eop)


    doInclinationAnalysis()

    doDebrisOnlyInclinationAnalysis()

    doDebrisAltitudeBinAnalysis()

    doDebrisSpacialDensityAnalysis()

    doProportionDebrisByAltitude()

    doCountDebrisNonDebrisComparisonByAltitude()

    calculateBandedCollisionFrequency()

    calculateBandedCollisionFrequencyDebToNonf()

    calculateBandedCollisionFrequencyNonftoNonf()
end

buildAllFigures()
```

Appendix B

All of the code used in the final prediction model:

```
using SatelliteToolbox
using Random
using Plots
using StatsBase
using LaTeXStrings
using DelimitedFiles

global eop = get_iers_eop()
# The DCM (Direction Cosine Matrix) that rotates
# TEME into alignment with ITRF
global D_ITRF_TEME = rECItoECEF(TEME(), ITRF(),
            DatetoJD(2022, 1, 1, 0, 0, 0), eop)
gr()

function buildModel()
    tles = read_tle("TLEData.txt")

    tles_DEBRIS = filter(e -> occursin("DEB", e.name), tles) #Debris
    tles_NOD = filter(e -> !occursin("DEB", e.name), tles)
# Not Debris
```

```
    ALL_altitudes = getTleAltitude.(tles)
    DEB_altitudes = getTleAltitude.(tles_DEBRIS)
    NOD_altitudes = getTleAltitude.(tles_NOD)

    ALL_Comb = [tles ALL_altitudes]
    ALL_Comb = ALL_Comb[(ALL_Comb[:, 2].<2500).&(ALL_Comb[:, 2].>100), :]

    tles = ALL_Comb[:, 1]
    ALL_altitudes = ALL_Comb[:, 2]

    DEB_Comb = [tles_DEBRIS DEB_altitudes]
    DEB_Comb = DEB_Comb[(DEB_Comb[:, 2].<2500).&(DEB_Comb[:, 2].>100), :]

    tles_DEBRIS = DEB_Comb[:, 1]
    DEB_altitudes = DEB_Comb[:, 2]

    NOD_Comb = [tles_NOD NOD_altitudes]
    NOD_Comb = NOD_Comb[(NOD_Comb[:, 2].<2500).&(NOD_Comb[:, 2].>100), :]

    tles_NOD = NOD_Comb[:, 1]
    NOD_altitudes = NOD_Comb[:, 2]

    TLEResults = [analyzeTLE.(tles_NOD, Ref(tles_DEBRIS),
                              Ref(tles_NOD), Ref(ALL_altitudes),
                              Ref(DEB_altitudes),
                              Ref(NOD_altitudes))][1]

    TLEResults = [getproperty.(tles_NOD, :name)
                    mapreduce(permutedims, vcat, results)]

    @show TLEResults

    TLERankings = [TLEResults[:, 1] ordinalrank(TLEResults[:, 2])
                    ordinalrank(TLEResults[:, 3])]
    TLERankings = [TLERankings (TLERankings[:, 2] .+ TLERankings[:, 3])]

    TLERankings = sortslices(TLERankings, by = (x) -> x[4], dims = 1)

    TLERankings[:, 1] .= chop.(TLERankings[:, 1], head = 2, tail = 0)

    TLERankings = [["Satellite" "P Rank" "Alpha Rank" "Total Score"]; T
                    LERankings]

    @show TLERankings
    writedlm("ModelOutput.csv", TLERankings, ",")
    writedlm("Top100ModelOutput.csv", TLERankings[1:102, :], ",")
end

# Helper Functions

function getTleAltitude(tle)
    orbp = init_orbit_propagator(Val(:sgp4), tle)
    try
```

```
        r_teme, v_teme = propagate_to_epoch!(
                        orbp, DatetoJD(2022, 1, 12, 0, 0, 0))
        r_itrf = D_ITRF_TEME * r_teme
        lat, lon, h = ecef_to_geodetic(r_itrf)

        return h / 1000
    catch
        return 0
    end
end

function getSpatialDensity(radius, count,  A )
    outerVolume = 4 / 3 *    * radius^3
    innerVolume = 4 / 3 *    * (radius -  A )^3

    sliceVolume = outerVolume - innerVolume

    return count / sliceVolume
end

function getRunawayThreshold(altitude, ALL_altitudes)
    bins = collect(0:50:2500)

    ALL_binned = fit(Histogram, ALL_altitudes, bins, closed = :right)

    CUMULATIVE = cumsum(reverse(ALL_binned.weights))

    popat!(bins, 1)


    Re = 6371 # radius of the earth, km
    G = 6.673 * 10^-11 # Gravitational constant
    Me = 5.972 * 10^24 # Mass of earth, Kg
    Cd = 2.2
    W = 1.5

    mA = 125 # Average mass over area - given by Kessler
    V = 7.5 # Avg. Relative velocity, km/sec, given by Kessler
    sigmaf = 14

    rNi_arr = (4 .* pi .* (Re .+ bins) .^ 3 .*
    .sqrt((G + Me) ./ (Re .+ bins)) .* 10^-11 .* Cd)
    ./ (CUMULATIVE .* W .* mA .* V .* sigmaf)

    return rNi_arr[Int(fld(altitude, 50))]

end

function analyzeTLE(tle, tles_DEBRIS, tles_NOD,
                    ALL_altitudes, DEB_altitudes, NOD_altitudes)

    tle_alt = getTleAltitude(tle)
    alt_band = 5 # 10 km band, 5 on either side
```

```
    Vs = 7 #km / s
    Ac = 4 #m^2

    Qd = length(filter((x) -> (x .> (tle_alt - alt_band) && x .<
                        (tle_alt + alt_band)), DEB_altitudes))
    Qs = length(filter((x) -> (x .> (tle_alt - alt_band) && x .<
                        (tle_alt + alt_band)), NOD_altitudes))

    Sd = getSpatialDensity(tle_alt + alt_band, Qd, alt_band * 2)
    Ss = getSpatialDensity(tle_alt + alt_band, Qs, alt_band * 2) * 0.35
    Ps = (Sd + Ss) * Vs * Ac

    ttl_objs = Qd + Qs
    sigmad = Qd / ttl_objs
    sigmas = Qs / ttl_objs

    Xd = 400
    Xs = 1000

    Di = Xd * sigmad + Xs * sigmas

    Dp = Di * Ps

    DeltaS = getSpatialDensity(tle_alt + alt_band,
            Qd + Dp, alt_band * 2) - Sd

    outerVolume = 4 / 3 * pi * (tle_alt + alt_band)^3
    innerVolume = 4 / 3 * pi * (tle_alt + alt_band - 10)^3

    DeltaV = outerVolume - innerVolume

    DeltaPf = 1 / 2 * DeltaS^2 * Vs * Ac * DeltaV

    rNi = getRunawayThreshold(tle_alt, ALL_altitudes)

    alpha = Dp / rNi

    return [DeltaPf, alpha]
end

buildModel()
```

Appendix C

Raw model output of the 100 highest risk satellites to the orbital environment.

Satellite	P Rank	Alpha Rank	Total Score
MAGION 2	1	4	5
DELTA 1 R/B	2	10	12
SL-12 R/B(AUX MOTOR)	3	11	14
COSMOS 1305	4	14	18
DELTA 1 R/B(1)	5	17	22
SCOUT X-4 R/B	6	18	24

Satellite	P Rank	Alpha Rank	Total Score
DELTA 1 R/B(1)	7	19	26
SL-12 R/B(AUX MOTOR)	8	20	28
BLOCK DM-SL R/B	9	22	31
CZ-2C R/B	10	29	39
ATLAS CENTAUR R/B	11	31	42
ALOUETTE 2	12	34	46
GLOBALSTAR M054	16	36	52
DELTA 2 R/B(1)	13	40	53
VANGUARD 1	14	41	55
INTERCOSMOS 25	15	45	60
PEGASUS R/B	17	54	71
GLOBALSTAR M030	25	46	71
SL-8 R/B	23	49	72
GLOBALSTAR M032	18	58	76
VANGUARD 3	32	47	79
GLOBALSTAR M053	20	63	83
OV3-3	19	75	94
SECOR 5 (EGRS 5)	21	82	103
GLOBALSTAR M036	48	55	103
GLOBALSTAR M026	22	88	110
GLOBALSTAR M006	37	76	113
OV1-14	35	79	114
GLOBALSTAR M044	52	64	116
ATLAS CENTAUR R/B	26	91	117
GLOBALSTAR M063	53	65	118
GLOBALSTAR M025	24	95	119
RADIO ROSTO	27	93	120
SL-19 R/B	28	100	128
GLOBALSTAR M035	29	102	131
SL-12 R/B(AUX MOTOR)	30	104	134
GLOBALSTAR M052	31	106	137
GLOBALSTAR M043	33	105	138
SCOUT X-1 R/B	44	101	145
DELTA 1 R/B(1)	38	107	145
GLOBALSTAR M048	36	110	146
DELTA 1 R/B(1)	51	96	147
MAGION 3	39	108	147
GLOBALSTAR M014	49	98	147
SL-12 R/B(2)	34	113	147
GLOBALSTAR M015	45	103	148
DELTA 2 R/B	40	109	149
SL-8 R/B	60	99	159
DFH-1	47	120	167
COSMOS 2356	57	123	180
OPTUS A1 R/B(PAM-D)	58	124	182
ARIANE 5 R/B	43	148	191
GLOBALSTAR M027	41	151	192
DIAMANT R/B	42	152	194
USA 16	65	131	196
GLOBALSTAR M024	46	155	201
USA 25	63	150	213
N-2 R/B(2)	64	154	218
GLOBALSTAR M051	59	170	229

Satellite	P Rank	Alpha Rank	Total Score
COSMOS 482 DESCENT CRAFT	54	179	233
GLOBALSTAR M061	68	165	233
USA 17	69	166	235
COSMOS 2392	55	180	235
ATLAS F R/B	62	175	237
SCOUT B R/B	71	168	239
GLOBALSTAR M049	50	192	242
M-4S R/B	73	181	254
DIAPASON (D1-A)	66	207	273
PEGASUS R/B	56	220	276
GLOBALSTAR M047	61	219	280
EXPLORER 25 (INJUN-4)	67	222	289
COSMOS 839	78	212	290
RADIO 8	79	221	300
GLOBALSTAR M002	70	231	301
COSMOS 1045	75	236	311
SL-8 R/B	76	237	313
SL-8 R/B	80	234	314
GLOBALSTAR M034	74	250	324
COSMOS 2352	72	253	325
SL-8 R/B	81	249	330
GLOBALSTAR M064	77	261	338
RADIO 3	83	263	346
EXOS D (AKEBONO)	84	266	350
SL-8 R/B	85	275	360
GLOBALSTAR M046	96	264	360
SL-8 R/B	97	265	362
DELTA 1 R/B(1)	86	285	371
TOPEX/POSEIDON	148	223	371
SL-8 R/B	87	294	381
COSMOS 341	153	232	385
TIROS 9	154	233	387
DELTA 2 R/B	156	235	391
GLOBALSTAR M058	82	311	393
SL-8 R/B	103	295	398
S6 MICHAEL FREILICH	158	245	403
JASON 3	159	246	405
SL-8 R/B	88	318	406
THOR DELTA 1 R/B	157	251	408
RADIO 4	95	321	416
DELTA 2 R/B(1)	168	252	420
COSMOS 539	164	257	421

References

[1] Kessler, D.J., Cour-Palais, B.G.: Collision frequency of artificial satellites: The creation of a debris belt. Journal of Geophysical Research: Space Physics **83**(A6), 2637–2646 (1978). https://doi.org/10.1029/JA083iA06p02637

[2] Kessler, D.J.: Collisional cascading: The limits of population growth in low earth orbit. Advances in Space Research **11**(12), 63–66 (1991). https://doi.org/10.1016/0273-1177(91)90543-S

[3] Instability of the present leo satellite populations. Advances in Space Research **41**(7), 1046–1053 (2008). https://doi.org/10.1016/j.asr.2007.04.081

[4] Component, U.S.S.C.J.F.S.: Space Strategic Awareness (SSA) Element Set (2022)

[5] Boley, A.C., Byers, M.: Satellite mega-constellations create risks in low earth orbit, the atmosphere and on earth. Scientific Reports **11**(1) (2021). https://doi.org/10.1038/s41598-021-89909-7

[6] Agency, E.S.: The current state of space debris. Accessed Jan.12, 2022 (2020). https://www.esa.int/Safety_Security/Space_Debris/The_current_state_of_space_debris

[7] Program, N.O.D.: Space Debris and Human Spacecraft. Accessed Jan. 12, 2022 (2021). https://www.nasa.gov/mission_pages/station/news/orbital_debris.html

[8] Program, N.O.D.: Orbital Debris Mitigation Standard Practices, November 2019 Update. https://orbitaldebris.jsc.nasa.gov/library/usg_orbital_debris_mitigation_standard_practices_november_2019.pdf. Accessed Jan. 24, 2022 (2019)

[9] Safety, E., Security: ESA commissions world's first space debris removal. https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal. Accessed Jan. 24, 2022 (2019)

[10] Center, G.S.F.: Satellite Safety. https://satellitesafety.gsfc.nasa.gov/. Accessed Jan. 24, 2022 (2021)

[11] Mann, A., Pultarova, T.: Starlink: SpaceX's satellite internet project. https://www.space.com/spacex-starlink-satellites.html. Accessed Feb. 22, 2022 (2022)

[12] Raju, N.: Russia's anti-satellite test should lead to a multilateral ban. https://www.sipri.org/commentary/essay/2021/russias-anti-satellite-test-should-lead-multilateral-ban. Accessed Feb. 22, 2022 (2021)

[13] David, L.: Avoiding satellite collisions: Noaa unveils prototype warning system. Space

[14] Weeden, B.: 2009 Iridium-Cosmos Collision Fact Sheet. https://swfound.org/media/6575/swf_iridium_cosmos_collision_fact_sheet_updated_2012.pdf. Accessed Feb. 22, 2022 (2010)

[15] Rao, R.: Avoiding satellite collisions: Noaa unveils prototype warning system. Space. Accessed 2022-02-22

[16] NASA: Frequently Asked Questions: Orbital Debris. https://www.nasa.gov/news/debris_faq.html. Accessed Feb. 22, 2022 (2011)