Five Stars, Must Read!: Clustering Analysis of Amazon Kindle Reviewers

Colin Petersen   
INFSCI0510  
*University of Pittsburgh*Pittsburgh, United States  
cpp18@pitt.edu

*Abstract*—In this analysis, I use agglomerative (bottom up) hierarchical clustering to create interesting groups of Amazon Kindle reviewers. The overall goal is to identify meaningful clusters that may represent bots, power users, irate reviewers, and more. These groupings have various implications and further uses for both Amazon corporate and Amazon customers. In my analysis, I investigate the clusters that result from a few different cutoff points through visual analysis of a dendrogram. In the end, I find clusters that may identify bots, negative reviewers, and positive reviewers, but further investigation is needed to correctly label each cluster and potentially implement decision rules to identify classes of users.

Keywords—Clustering, reviews, Amazon reviews, Kindle, hierarchical clustering, agglomerative clustering, bot detection

# Introduction

Amazon reviews are extremely important to both consumers and business owners. Bot detection service Fakespot estimated that 42% of the 720 million Amazon product reviews they assessed in 2020 were fraudulent.[1] Additionally, consulting group the Behaviouralist estimates that fake reviews cost consumers 12 cents for every dollar they spend.[2] The incentive to fabricate positive reviews is worth the risk for some sellers: the e-commerce consultant Pattern found that a one-star increase on a product could increase sales of that product by as much as 26%.[2] The stakes are high to combat botting.

Outside of bot detection systems, it may be useful to flag users as ones that tend to leave positive or negative reviews. If “Negative Nancy” reviews 10 products with a 1-star review and 1 product with a 5-star review, that 5-star review might mean more than another 5-star review on the same product from a user that only leaves positive reviews. Savvy consumers could benefit from knowing whether a reviewer is from a quality contributor, and Amazon could benefit by identifying bots to delete and power users to potentially reward. Consumers put a lot of trust into online strangers for purchases at every price point. Knowing which reviewers to trust and which to take with a grain of salt can lead to smarter purchases, less wasted money, and a greener earth with less carbon emissions that come from the fuel required to ship returns and the industrial output that comes from the production of unnecessary products.

Kindle reviews are a particularly interesting category, because many Kindle eBooks are self-published. Reviews in this space can have massive impacts on indie authors, both good and bad. Saoud Khalifah, the founder of Fakespot, identifies books as one of the most fraud-proof products, because “You cannot fake a really detailed review talking about a book.”[2]

My data comes from a 2018 scrape by UCSD. The data is the “5-core Kindle Review Data” that contains Amazon reviews from accounts with 5 or more reviews on Kindle products with 5 or more reviews.[3] In this analysis, I clean the data, aggregate the data so each row represents a reviewer and not an individual review, then I scale the data, implement my clustering algorithms, and investigate my models.

# Methodology

## Data Story: Cleaning and Descriptions

I first started out by cleaning my initial dataset to remove duplicates, NA values that were errors and/or less than 5% of my total data, and potentially not applicable products. After this, I was left with 2,141,350 observations. For each individual review, I also took the text data of the review header and the review body and counted the number of characters in each. From these observations, I aggregated the data by unique user ID, and created the features I plan to use in my data analysis. Now, each row represents all Kindle reviews for a particular user. From now on, a “review” refers to a review of a Kindle product listed in this dataset. The features I will use to make my clusters are:

* unixReviewTime\_min\_gap: The smallest gap between review postings in seconds. For example, if a user posted reviews on 12/12/15, 12/11/15, 2/6/15, and 4/18/15, the smallest gap would be the gap between 12/12/15 and 12/11/15, which is one day or 86400 seconds. If two reviews are posted on the same day, this gap is zero.
* Overall\_max: The maximum rating a user has given a product (1, 2, 3, 4, or 5)
* Overall\_min: The maximum rating a user has given a product (1, 2, 3, 4, or 5)
* Overall\_mean: The mean of all ratings this user has given on various products (between 1 and 5)
* reviewLength\_max: The length in characters of the longest review a user has posted.
* reviewLength\_min: The length in characters of the shortest review a user has posted.
* reviewLength\_mean: The average length in characters of all of a user’s reviews.
* summaryLength\_max: The length in characters of the longest review summary a user has posted.
* summaryLength\_min: The length in characters of the shortest review summary a user has posted.
* summaryLength\_mean: The average length in characters of all of a user’s review summaries.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Variable: | | | | | | |
| unixReviewTime\_min\_gap | overall  \_max | overall  \_min | overall  \_mean | reviewLength\_max | reviewLength\_min | reviewLength\_mean |
| mean | 7.201409e+05 | 4.932577 | 3.219222 | 4.376642 | 831.58277 | 142.60853 | 390.01113 |
| std | 1.964542e+06 | 0.289394 | 1.358548 | 0.642118 | 1075.9653 | 213.18415 | 463.68330 |
| min | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Q1 | 0 | 5 | 2 | 4 | 244 | 38 | 141.60000 |
| Q2 | 0 | 5 | 3 | 4.566667 | 469 | 104 | 237.31414 |
| Q3 | 5.184000e+05 | 5 | 4 | 4.888889 | 974 | 150 | 454.80000 |
| max | 4.164480e+07 | 5 | 5 | 5.000000 | 30000 | 25717 | 25760.333 |

* Vote\_sum: The sum of all “helpful” votes that a user has received across their reviews.
* reviewerID\_count: The number of reviews this user has posted.
* Verified\_mean: The percentage of a user’s reviews where the user is labeled as a verified purchaser of the product.

Initially, I thought that the Unix review time would be accurate to the second, so I figured that the minimum gap in reviews could be used to identify bots by looking at impossible gaps between reviews. Unfortunately, it is only accurate to the day posted.

Overall min, max, and mean can tell us something about a user’s tendencies in leaving reviews. If a user with 20 reviews has a max of 3, a mean of 1.4, and a min of 1, they may tend to be a more negative reviewer who only leaves a review when they are upset. A user with 20 reviews with a max of 5, a mean of 5, and a min of 5 only leaves 5 star reviews. In general, people tend to leave 5-star or 4-star reviews.

Review length min, max, and mean can tell us how detailed users are in their reviews. If a user’s max review is short, they are never detailed. If a user’s min review is long, they are always detailed. The mean review length can tell us about a user’s tendency to write long reviews. Summary length min, max, and mean tell us the same thing but for the review summary (like the header or title of the review). Perhaps power users tend to write long reviews with short, descriptive, and eye-catching summaries.

Vote\_sum can say a few different things about a user. It does not always indicate how helpful their responses typically are, but it can in some cases. A great, detailed review that shares novel information that will inform a buyer may mean that 1 out of every 1000 shoppers find the review helpful. Maybe a terrible review that is not useful will be found helpful by 1 out of 50,000 shoppers due to missclicks. Maybe a great review is only viewed 100 times and is never marked as helpful, and a review of similar quality on a more active page is viewed 10,000 times and receives 10 helpful votes. A helpful vote can say things about how much a review was viewed and/or how helpful the review was.

Finally, verified\_mean may be able to identify bots or fraudulent reviewers. reviewerID\_count puts all of these other variables in context and can identify power users. A user with 7 reviews that are all verified with 500 total helpful votes is probably a higher quality contributor than a person with 500 unverified reviews with no helpful votes.

After aggregating by reviewer, I removed all users that had less than five reviews. I did this because it would be difficult to classify a person with just one review, but it is helpful to leave the cutoff low so I can generalize the classification groups to more people. My problem’s scope becomes different if I am classifying users that have 100 reviews or more. This is not exactly a problem, but also not what I set out to do. I will go into more detail on final variable selection and appropriate scaling in later sections.

## Descriptive Statistics

1. Descriptive Statistics of the first seven variables

In table I above, we observe a few interesting things. At least half of all reviewers have written two reviews on the same day. The mean user’s mean review is quite high at 4.376642. At least 75% of all reviewers in this dataset have given a five-star review, and at least 50% of all reviewers have not given a score lower than 3 stars. Length is strongly right skewed with the max length at the character limit, a mean user mean of 390.01113, and a median user mean of 237.31414.

1. Descriptive Statistics of the Next six variables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Variable: | | | | | |
| summaryLength\_max | summaryLength\_min | summaryLength\_mean | vote\_sum | reviewerID\_count | verified\_mean | |
| mean | 48.869080 | 9.580819 | 23.264690 | 9.722862 | 15.996987 | 0.718822 | |
| std | 28.426017 | 5.974380 | 12.000780 | 34.947817 | 29.880954 | 0.326982 | |
| min | 2 | 1 | 1.857143 | 0 | 5 | 0.000000 | |
| Q1 | 28 | 6 | 14.600000 | 0 | 6 | 0.500000 | |
| Q2 | 44 | 9 | 20.629384 | 2 | 8 | 0.846154 | |
| Q3 | 64 | 11 | 28.920769 | 8 | 14 | 1.000000 | |
| max | 728 | 128 | 160.600000 | 2546 | 1366 | 1.000000 | |

Table II with the rest of the variables has a few other results to note. At least 50% of all reviewers that have reviewed at least 5 Kindle books have reviewed 8 books or less, but one user has reviewed 1366 books. Summary lengths, as opposed to review lengths, are a lot less right skewed. The number of Kindle books reviewed by the mean Kindle reviewer is about double the median. Finally, at least 25% of all users have verified their purchase in all their reviews, and at least 75% of users have verified 50% or more of their purchases in their reviews.

## Model Creation and Justification

The method I will use to create clusters is agglomerative clustering. My decision to choose agglomerative clustering over divisive clustering mostly boils down to the ease of implementation in Python. Perhaps in a future investigation (especially if scalability is a concern), a divisive clustering model that is not built out to its full depth when constructing an initial dendrogram could be implemented to save memory and time.

As a hierarchical clustering model, agglomerative clustering has a few distinct advantages over other models like K-means. K-means assumes a spherical cluster shape, which may not exist in the highly skewed and noisy data. Hierarchical clustering does not assume any particular cluster shape. K-means is also nondeterministic, meaning that as K-means is repeated, even with the same number of clusters K, it will yield a different result each time due to the random assignment of starting points. Hierarchical clustering is deterministic, which allows analysis to be reproducible as long as the data stays the same.

Agglomerative clustering also has the advantage of being more robust and easier to understand when choosing the number of clusters (which I will define as c for the rest of the paper). Building a full depth dendrogram allows one to visually inspect what level of granularity corresponds to what number of clusters and how distant each cluster is from a previous split at each increase in c. This allows for an effective balance of detail and distance. There’s no use having 100 clusters if there are 10 groups of 10 clusters that are all highly similar. This would lead to many clusters that give relatively the same information as other clusters. Similarly, there’s no use in having only 2 clusters if the goal of clustering is to segment into multiple highly specific groups.

The main drawback of agglomerative clustering is the intense computation expense as well as the large amount of memory required. Indeed, in this analysis, I had to use a random subset of 10% of my data to build the dendrogram to identify cutoff points and then 30% of the data to build the clustering model on 3 different levels of c. Additionally, outliers can have a large effect on the final model, which may be a concern considering the extreme skew of my predictors.

To evaluate my models and choose the “best” one, I will use silhouette score as my evaluation metric. Silhouette score is a measure of (mean intercluster distance - mean intracluster distance)/max(mean intracluster distance, mean intercluster distance). It takes on values from -1 to 1, with 1 being the ideal score. As silhouette score approaches 1, clusters become denser and more well-defined with a high degree of separation.[4]

However, having the highest quality clusters by silhouette score may not satisfy the overall goal of this analysis, which is to identify interesting groups, particularly ones that could help Amazon corporate (bots, power users) and ones that could help individuals (balanced reviewers, positive reviewers, negative reviewers, etc). Granularity is important to me, because some of these groups may be a small percentage of the overall data. Too much granularity will be overwhelming, however. I want to see how groups evolve and break apart as the number of clusters increases. I also want to take a model that balances granularity, interpretability, and silhouette score and try to identify patterns that would be noteworthy to Amazon corporate and Amazon shoppers.

Before creating the models and choosing split points, the data needs to be scaled. Due to the highly skewed nature of the data and the fact that there doesn’t appear to be any well-defined distribution underlying the data, a minmax scaling function will be implemented. Because many variables are set within well-defined bounds where most of the data lies up against a boundary (for example, overall\_max cannot exceed 5, which is its first quartile), normalization or standard scaling would fall flat.

After creating the models, I dig deeper into the decisions by visualizing the clusters. First, to visually represent my clusters in two dimensions, I will do PCA with two principal components. PCA (or principal component analysis) is a dimension reduction technique that turns variables into combinations of variables meant to maximize variance. Using PCA, I can represent each cluster as a color on a 2d scatter plot where the X value is principal component 1 and the Y value is principal component 2. After doing this, I output side-by-side boxplots for each variable that are separated by cluster. This allows me to attempt to label each cluster and see what makes each cluster different.

# Results

I first conducted this analysis while including various minimums and maximums as well as means for variables like summaryLength, reviewLength, and overall. Upon further investigation, means, minimums, and maximums are highly correlated, while most mins and maxes (except for overall\_min) had a low standard deviation after scaling and didn’t seem to be very useful. I also removed unixReviewTime\_min\_gap, as it was sort of a failed experiment due to the fact that the unix review time is only accurate to the day. After doing that, I was left with overall\_min, overall\_mean, reviewLength\_mean, summaryLength\_mean, vote\_sum, reviewerID\_count, and verified\_mean.

Chart, histogram

Description automatically generated

1. Dendrogram used to find optimal number of clusters to try.

Using a 10% subset of the newly scaled and pruned data, I constructed this dendrogram in figure 1 above using the default distance metric of Euclidean distance and the Ward linkage method. I used a subset of only 10% of my data due to the huge memory and time complexity of the algorithm given my number of observations (132,432) and my number of predictors (7). From here, I wanted to capture various levels of cluster distance. First and foremost, it seems that a lot of the distance comes from c=3, which corresponds to a cutoff point of about y=30. Next, I decided to choose a cutoff point of approximately y=16, corresponding to 6 clusters. This is because after this point, distances between clusters from splits begin to decrease quickly. Finally, I choose a cutoff point at approximately y=11, corresponding to 9 clusters for added granularity. Finally, I chose an arbitrary value of 20 clusters to see if the increased density of clusters offsets the increase in intracluster distance. I will not display any graphical representations of c=20, because it is too difficult to see the differences between clusters.

After that, I used my 30% subset to construct my clustering models. Table III below represents my silhouette scores for different c values. C=3 had the highest silhouette score, and 9 clusters had the second highest score. Ultimately, to achieve the goals of my analysis, I want to deviate from diving deeper into the model with the highest silhouette score and instead focus on the model with the second highest score, the model with c=9. Although it is not the best, it strikes the best balance of silhouette score and granularity. Interestingly, the model with 20 clusters performed better than the model with only 6, although both were worse than c=9 and c=3.

1. Silhouette Scores By Number of Clusters

| C | Silhouette Score |
| --- | --- |
| 3 | 0.3750914319079 |
| 6 | 0.3320475282795 |
| 9 | 0.3455618720296 |
| 20 | 0.3431780877906 |

After doing PCA on my features, I plotted my clusters in 2D. Figures 2, 3, and 4 below show the results.

Chart

Description automatically generated

1. PCA representation of clustering with c=3. The X axis represents the first principal component and the Y axis represents the second.

Chart

Description automatically generated

1. PCA representation of clustering with c=6. The X axis represents the first principal component and the Y axis represents the second.

Chart

Description automatically generated

1. PCA representation of clustering with c=9. The X axis represents the first principal component and the Y axis represents the second.

Within each level of c, there are some irregular cluster shapes and some overlap between clusters. This is interesting and could potentially be invested further using 3 principal components and some package that allows for interactive 3D modeling. Additionally, PCA has created 4 distinct “islands” of observations that get thicker from left to right. Some clusters tend to sprawl across islands, and some tend to spread up and down on one island. The evolution as c increases is interesting as well. The largest island is initially all cluster 0 in figure 2, but by figure 4, there are 4 clusters that inhabit the island. Cluster 0 is by far the largest cluster by number of observations in figure 2. This is interesting because it gives me an idea of the most typical Amazon reviewer, but it does not give me the granularity I want to potentially find bots, power users, etc. Comparing the model where c=6 to the model where c=9, we see an even greater segmentation of the far right “island” where a large portion of the data lies. Given that the problem of overlap between clusters seems to get worse from c=6 to  c=9, the increase in silhouette score from c=6 to c=9 likely comes from an increase in mean intercluster distance that offsets the poor separation.

Now, to evaluate the boxplots. First, across models c=3, c=6, and c=9, it appears that the cluster with the lowest verified\_mean tends to have some of the highest overall\_mean. This shows that the reviewers for whom verified purchases make up a very low percentage of their total reviews tend to rate products positively, potentially indicating bot activity. In the evolution from c=3 to c=9, the large rightmost “island” was partitioned into 4 clusters, which segment what was originally cluster 0, mostly based on verified\_mean, overall\_min, and overall\_mean. I did not include the boxplots for all models in this paper, as it would take up too much space.

Finally, I want to look at specific clusters in my final model and give my best guess as to what type of reviewer they indicate. The figures below represent variables that are of particular interest.

Chart, box and whisker chart

Description automatically generated

1. Side-by-side boxplots of overall\_min at cluster levels 0 through 8.

Chart, box and whisker chart

Description automatically generated

1. Side-by-side boxplots of overall\_mean at cluster levels 0 through 8.

Chart, box and whisker chart

Description automatically generated

1. Side-by-side boxplots of reviewLength\_mean at cluster levels 0 through 8.

Chart, box and whisker chart

Description automatically generated

1. Side-by-side boxplots verified\_mean at cluster levels 0 through 8.

I will now attempt to create my own labels for some select clusters. First, I will define cluster 7 as the “positive Pollys.” This cluster is highly verified, and never gives out anything lower than 5 stars. They also tend to keep their reviews short and sweet. These people may refuse to give out poor reviews to products they don’t like. They also may be highly intelligent shoppers who only purchase eBooks that they end up genuinely enjoying.

Interestingly, clusters 1 and 6 were the least verified clusters, but they tended to write some of the longest reviews. Cluster 6 was not afraid to give out negative reviews, while cluster 1 tended to be more positive. It’s interesting and a little distressing that the clusters full of users who wrote the longest reviews also tended to be the least verified. Are these accounts bots? Does this mean that bots tend to write more than humans do, giving the illusion of a well-thought out response? Why write so much and not bother to verify your purchase? I suppose it wouldn’t be too hard for an AI to analyze an eBook and write a long review praising it, even in 2018. Additionally, cluster 6 has the highest reviewerID\_count by far, while being nowhere near the largest cluster of the 9. I will call these clusters the “potential bots” (although more analysis is needed– perhaps the positive Pollys are all bots that cost extra for verified reviews).

Finally, we have cluster 8, the “negative Nancys.” Everyone in this cluster (except for one outlier) has given a one-star review at least once, and the median of their average ratings is just 3 stars, well below the overall median of all reviewers. The negative Nancys tended to write a little more than the positive Pollys. This makes sense for eBook reviews. Reading is a time investment, and when a reader doesn’t like a book, there’s usually a specific reason or number of reasons why. Liking a book may be more subconscious or hard to put into words. Perhaps some of these users never leave a review when they like a book but always leave a review when they hate it. Maybe these users are hard to please, or maybe they’re terrible at guessing what type of books they like before they buy.

Clusters 2, 3, and 4 are relatively similar across most metrics. Clusters 0 and 5 are more verified than the potential bots but less verified than the rest of the clusters. Although I intended to discover a category of “power users” or people who write a lot of quality reviews, through further investigation I determined that the definition of a “power user” is too hard to nail down, and because they make up such a small segment of Amazon users, they are difficult to pin to one cluster.

Additional figures will be included in the appendix. It’s interesting to note the emergent properties of each new cluster as they split off from previous clusters. Cluster 1 in the c=6 model split into cluster 8 and 2 in the final model. In the model with c=6, cluster 1 had a median scaled overall\_mean of 0.7. This is because what would later become cluster 2 was masking the low overall\_mean from the “negative Nancy” group. Clusters 2 and 8 in the final model were likely lumped together in the c=6 model because they share a low overall\_min, even though cluster 2 is not as consistently negative as cluster 8.

# Discussion

## Summary

In this analysis, I found that c=3 clusters is the best clustering split by means of silhouette score. Even so, the silhouette score of ~0.3751 is not great. More textual analysis through natural language processing could be useful. Perhaps complexity of language, some sort of grammar score, and some sort of sentiment analysis could be useful. I would hesitate to extend any of the results beyond the population analyzed, which was Amazon Kindle reviewers on eBook products where both the product and reviewer have 5 or more reviews.

Nine clusters were the best balance between granularity, interpretability, and silhouette score. If I had a more powerful computer, I would try all values from 6 to 15 to see which has the highest silhouette score. Within those nine clusters, I identified what I believe to be different groups of reviewers. Further study is needed to verify those results, but it’s an interesting starting point.

## Limitations and caveats

A limitation to implementing this model in practice would be the huge time and memory complexity of constructing dendrograms and agglomerative clustering models. Additionally, I was limited by not knowing the true labels of reviewers. Overall, I’m not sure if this analysis in its current state would be truly useful for consumers or Amazon employees.

Because of the high dimensionality of the data and the subjective nature of my cluster labels, it is very difficult to evaluate how accurate my assumptions are. The easiest one to validate would be whether or not someone is a bot, if I had a subset of userIDs that are known bots. From there, I could use additional metrics like Rand’s index[5] to see what number of clusters effectively sniff out bots from humans.

Additionally, it is possible that my random 30% sample missed some important data points that may help define new clusters, clarify the boundaries between groups, and increase the density of existing clusters. This is unavoidable due to the memory and time complexity involved in hierarchical clustering, although there may be some way to combat this that I didn’t think of.

## More Future Use Ideas

The overall goal of this analysis was to identify clusters that could potentially benefit Amazon consumers and/or Amazon corporate. Emphasis was placed on identifying certain types of reviewers. On the consumer side, this analysis could be used in the future to assist in the creation of a Google Chrome extension that identifies when reviewers are acting outside of the typical behavior for their cluster. Obviously, this algorithm would need some tweaking, the data would be hard to update in real time without issues, and constantly adding new data can make this quite expensive, challenging, and possibly illegal. Furthermore, the subject of this investigation was only a select subset of Kindle reviewers. Generalizing this information would be quite challenging.

More than anything else on the consumer side, this investigation should be seen as a tool for Kindle eBook purchasers to check their assumptions and familiarize themselves with how machine learning techniques classify reviewers that exist on Amazon. Was that long five-star review really written by a human? What can be said about the person writing this review? Do they sound like the type of person who’s easy to please? What are they saying in their review? Do the ratings for this eBook seem suspiciously good or bad without specific reasoning?

In terms of Amazon corporate, this analysis could be used to help identify potential bots and try to create a more intelligent star system that weighs the reviews of some users more than others. Textual analysis and NLP models could be implemented in support of this model. It’s very common to see one-star reviews that say something like “GREat product, grandkids love it” or “the package that this product came in got lost or was mishandled so now I’ll take my anger out on the product instead of the postal service.” Perhaps those people all tend to fall into one cluster, or perhaps someone could find a way to quantify the tendency to write misleading or inaccurate reports which could then be added as a factor.

Overall, I think those goals are quite lofty and do not add up to what this model is in its current state. I’m sure the data scientists working for Amazon could generate a model with a much higher silhouette score, more sensible features, and more interesting conclusions. If I was able to repeat my process with more high-quality covariates that contain a large amount of variance, perhaps I could make more effective and enlightening clusters.

##### References

1. S. Maheshwari. “When Is a Star Not Always a Star? When It’s an Online Review.” New York Times. https://www.nytimes.com/2019/1 1/28/business/online-reviews-fake.html
2. M. Stieb. “Amazon’s War on Fake Reviews.” New York Magazine. https://nymag.com/intelligencer/2022/07/amazon-fake-reviews-can-they-be-stopped.html
3. Jianmo Ni, Jiacheng Li, Julian McAuley  
   *Empirical Methods in Natural Language Processing (EMNLP)*, 2019 https://nijianmo.github.io/amazon/index.html#sample-review
4. A. Kumar. “KMeans Silhouette Score Explained With Python Example.” Dzone. <https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam>
5. Scikit-learn staff. “2.3. Clustering.” Sci-kit-learn.org https://scikit-learn .org/stable/modules/clustering.html
6. Various lecture slides from Dmitry Babichenko were used throughout as wel. Thanks for a great year!

.

**APPENDIX:**

Here are some tables and graphs that are helpful to have but too bulky to include in the main part of the analysis:

Table

Description automatically generated

(Above and below): Description of data after scaling

Table

Description automatically generated

Below: histogram and density plot of all factors used in final model after scaling

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Below: heatmap of correlations that was used to consider dropping multiple summary statistics for the same value

Chart, histogram

Description automatically generated

Below: Select Histograms from c=3 and c=6 that may be of interest

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Chart, waterfall chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated