**Introduction**

[Last week I analyzed player rankings of the Arkham Horror LCG classes.](https://wp.me/p27liZ-Vr) This week I explain what I did in the data analysis. As I mentioned, this is the first time that I attempted inference with [rank data](https://en.wikipedia.org/wiki/Ranking), and I discovered how rich the subject is. A lot of the tools for the analysis I had to write myself, so you now have the code I didn’t have access to when I started.

This post will *not* discuss rank data modelling. Instead, it will cover what one may consider basic statistics and inference. The primary reference for what I did here is [*Analyzing and Modeling Rank Data*, by John Marden](https://books.google.com/books/about/Analyzing_and_Modeling_Rank_Data.html?id=Dp24H21QHNoC). So far I’ve enjoyed his book and I may even buy a personal copy.

**What is Rank Data?**

Suppose we have mobjects we ask our study participants (also known as “judges”) to rank. For example, suppose we asked people to rank apples, oranges, and bananas. What we then get is a prioritization of these objects according to our judges. This could come in the form

(3, 1, 2)

and we interpret the number in the k^{th}position as the ranking of the k^{th}item. In this case, if the tuple is in the order of apples, oranges, and bananas, then oranges recieved the highest ranking, bananas the second-highest, and apples the last position.

An alternative view of this data may be

(\text{oranges}, \text{bananas}, \text{apples})

where the items are arranged in order of preference. This form of describing a ranking has its uses, but we will consider only the first form in this introduction.

Ranking data has the following distinguishing characteristics from other data: first, the data is ordinal. All that matters is the order in which items were placed, not necessarily the numbers themselves. We could insist on writing rank data as (100, 1, 50)and the information content would not have changed. (But of course we would never do this.) Second, every item gets a ranking. This excludes “Choose your top 3 out of 50”-type questions, since not every item would receive a ranking (this is called an incomplete ranking and requires special care; I won’t discuss this type of data in this article). Finally, every item’s ranking is distinct; no ties are allowed.

Thus ranking data is distinct even from just [ordinal data](https://en.wikipedia.org/wiki/Ordinal_data) since data comes from judges in the form of a tuple, not just a single ordinal value. (Thus we would not consider, say, [Likert scale responses](https://en.wikipedia.org/wiki/Likert_scale) as automatically being an instance of rank data.) An ideal method for rank data would account for this unique nature and exploit its features.

**Basic Descriptive Statistics**

From this point on I will be working with the Arkham Horror player class ranking data. I made the Timestamp column nonsense to anonymize the data. You can [download a CSV file of the data from here](https://docs.google.com/spreadsheets/d/1ksnE29BOgOVYbCOZqlB3unKKaLQ-7G6W5C_3x9yJinQ/edit?usp=sharing), then convert it to a .Rda file with the script below (which is intended to be run as an executable):

#!/usr/bin/Rscript

################################################################################

# ArkhamHorrorClassPreferenceSurveyDataCleaner.R

################################################################################

# 2019-02-10

# Curtis Miller

################################################################################

# This file takes a CSV file read in and cleans it for later analysis, saving

# the resulting data in a .Rda file.

################################################################################

# optparse: A package for handling command line arguments

if (!suppressPackageStartupMessages(require("optparse"))) {

install.packages("optparse")

require("optparse")

}

################################################################################

# MAIN FUNCTION DEFINITION

################################################################################

main <- function(input, output = "out.Rda", help = FALSE) {

input\_file <- read.csv(input)

input\_columns <- names(input\_file)

arkham\_classes <- c("Survivor", "Guardian", "Rogue", "Seeker", "Mystic")

for (cl in arkham\_classes) {

names(input\_file)[grepl(cl, input\_columns)] <- cl

}

names(input\_file)[grepl("Reason", input\_columns)] <- "Reason"

input\_file$Reason <- as.character(input\_file$Reason)

input\_file$Timestamp <- as.POSIXct(input\_file$Timestamp,

format = "%m/%d/%Y %H:%M:%S", tz = "MST")

for (cl in arkham\_classes) {

input\_file[[cl]] <- substr(as.character(input\_file[[cl]]), 1, 1)

input\_file[[cl]] <- as.numeric(input\_file[[cl]])

}

survey\_data <- input\_file

save(survey\_data, file = output)

}

################################################################################

# INTERFACE SETUP

################################################################################

if (sys.nframe() == 0) {

cl\_args <- parse\_args(OptionParser(

description = paste("Converts a CSV file with survey data ranking",

"Arkham Horror classes into a .Rda file with a",

"well-formated data.frame"),

option\_list = list(

make\_option(c("--input", "-i"), type = "character",

help = "Name of input file"),

make\_option(c("--output", "-o"), type = "character",

default = "out.Rda",

help = "Name of output file to create")

)

))

do.call(main, cl\_args)

}

(The script with all the code for the actual analysis appears at the end of this article.)

The first statistic we will compute for this data is the marginals matrix. This matrix simply records the proportion of times an item received a particular ranking in the sample. If we want to get mathematical, if \omegais a ranking tuple and \omega_iis the ranking of the i^{th}option and the sample is \omega^{(1)}, \ldots, \omega^{(n)}, then the ij^{th}entry of the marginal’s matrix is

\hat{M}_{ij} = \frac{1}{n} \sum_{l = 1}^n I_{\{\omega^{(l)}_i = j\}}

where the function $\latex I\_{{A}}$ is 1 if Ais true and 0 otherwise. (Thus the sum above simply counts how many times \omega^{(l)}_iwas equal to j.)

The marginals matrix for the Arkham Horror data is given below

MARGINALS

---------

1 2 3 4 5

Guardian 18.29 20.43 26.84 19.71 14.73

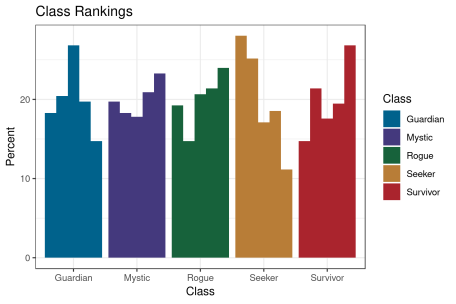
Mystic 19.71 18.29 17.81 20.90 23.28

Rogue 19.24 14.73 20.67 21.38 23.99

Seeker 28.03 25.18 17.10 18.53 11.16

Survivor 14.73 21.38 17.58 19.48 26.84

Below is a visual representation of the marginals matrix.



From the marginals matrix you could compute the vector representing the “mean” ranking of the data. For instance, the mean ranking of the Guardian class is the sum of the ranking numbers (column headers) times their respective proportions (in the Guardian row); here, that’s about 2.9 for Guardians. Repeat this process for every other group to get the mean ranking vector; here, the mean rank vector is (2.92, 3.10, 3.16, 2.60, 3.22)(keeping the ordering of the classes suggested by the rows above, which is alphabetical order; this will always be the ordering I use unless otherwise stated.) Of couse this is not a ranking vectors; rankings are integers. The corresponding ranking vector would be to rank the means themselves; this gives a ranking vector of (2, 3, 4, 1, 5).

I don’t like inference using the mean ranking vector. As mentioned above, this data is ordinal; that means the magnitude of the numbers themselves should not matter. We could replace 1, 2, 3, 4, 5 with 1, 10, 100, 1000, 10000 and the data would mean the same thing. That is *not* the case if you’re using the mean rank unless you first apply a transformation to the rankings. In short, I don’t think that the mean ranking vector appreciates the nature of the data well. And since the marginals matrix is closely tied to this notion of “mean”, I don’t think the matrix is fully informative.

Another matrix providing descriptive statistics is the pairs matrix. The matrix records the proportion of respondents who preferred one option to the other (specifically, the row option to the column option). Mathematically, the ij^{th}entry of the pairs matrix is



The pairs matrix for the Arkham Horror data is below:

PAIRS

-----

Guardian Mystic Rogue Seeker Survivor

Guardian 0.00 54.16 55.34 42.52 55.82

Mystic 45.84 0.00 51.07 39.90 53.44

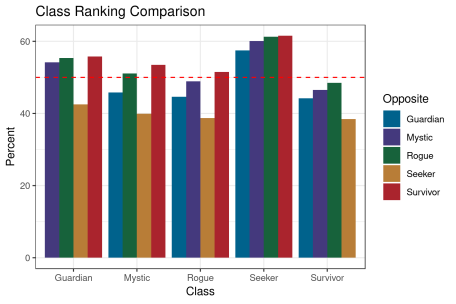
Rogue 44.66 48.93 0.00 38.72 51.54

Seeker 57.48 60.10 61.28 0.00 61.52

Survivor 44.18 46.56 48.46 38.48 0.00

First, notice that the diagonal entries are all zero; this will always be the case. Second, the pairs matrix is essentially completely determined by the entries above the diagonal of the matrix. Other forms of interence use these upper-diagonal entries and don’t use the lower-diagonal entries since they give no new information. The number of upper-diagonal entries is \bar{m} = m(m - 1)/2, which is the number of ways to pick pairs of mclasses.

The pairs matrix for the Arkham Horror data is visualized below.



With the pairs matrix, crossing above or below 50% of the sample being in the bin is a significant event; it indicates which classes are preferred to the other. In fact, by counting how many times this threshold was crossed, we can estimate that the overall favorite class is the Seeker class, followed by Guardians, then Mystics, then Rogues, and finally Survivors. This is another estimate of the “central”, “modal”, or “consensus” ranking. (This agrees with the “mean” ranking, but that’s not always going to be the case; the metrics can disagree with each other.)

While I did not like the marginals matrix I do like the pairs matrix; I feel as if it accounts for the features of rank data I want any measures or inference to take account of. It turns out that the pairs matrix is also related to my favorite distance metric for analyzing rank data.

**Distance Metrics for Rank Data**

A [*distance metric*](https://en.wikipedia.org/wiki/Metric_(mathematics)) is a generalized notion of distance, or “how far away” two objects xand yare. In order for a function d(x, y)to be a metric, it must have the following properties:

1. d(x, y) \geq 0for all xand y.
2. d(x, y) = 0if and only if x = y.
3. d(x, y) = d(y, x)for all xand y.
4. d(x, y) \leq d(x, z) + d(y, z)for all x, y, z(the “triangle  
   inequality”)

The notion of distance you use in every-day life, the one taught in middle-school geometry and computed whenever you use a ruler, is known as [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). It’s not the only notion of distance, though, and may not be the only distance function you use in real-life. For instance, [Manhattan or taxi cab distance](https://en.wikipedia.org/wiki/Taxicab_geometry) is the distance from one point to another when you can only make 90-degree turns and is the distance that makes the most sense when travelling in the city.

There are many distance metrics we could consider when working with rank data. The Spearman distance is the square of the Euclidean distance, while the footrule distance corresponds to the Manhattan distance. It turns out that the mean rank vector above minimizes the sum of [Spearman distances](https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient). The distance metric I based my analysis on, though, was the [Kendall distance](https://en.wikipedia.org/wiki/Kendall_tau_distance). I like this distance metric since it is not connected to the mean and considers the distance between the rankings (1, 2, 3, 4, 5)and (5, 2, 3, 4, 1)to be greater than the distance between (1, 2, 3, 4, 5)and (2, 1, 3, 4, 5)(unlike, say, the [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance), which gives the same distance in either case).

Kendall’s distance even has an interpretation. Suppose that two ranking tuples are seen as the ordering of books on a bookshelf. We want to go from one ordering of books to another ordering of books. The Kendall distance is how many times we would need to switch adjacent pairs of books (chosen well, so as not to waste time and energy) to go from one ordering to the other. Thus the Kendall distance between (1, 2, 3, 4, 5)and (2, 1, 3, 4, 5)is one; we only need to make one swap. The distance between (1, 2, 3, 4, 5)and (5, 2, 3, 4, 1), in comparison, is seven, since we need to make seven swaps.

It also turns out that the Kendall distance is related to the pairs matrix. The average Kendall distance of the judges from any chosen ranking \omegais

\sum_{i, j: \omega_i > \omega_j} \hat{K}_{i,j}

(There is a similar expression relating the Spearman distance to the marginal matrix.)

**Central Ranking Estimator**

Once we have a distance metric, we can define what the “best” estimate for the most central ranking is. The central ranking is the \omegathat minimizes

\frac{1}{n}\sum_{l = 1}^{n} d(\omega^{(l)}, \omega)

In other words, the most central ranking minimized the sum of distances of all the rankings in the data to that ranking.

Sometimes this ranking has already been determined. For instance, when using the Spearman distance, the central ranking emerges from the “mean” rankings. Otherwise, though, we may need to apply some search procedure to find this optimal ranking.

Since we’re working with rank data, though, it’s very tempting to not use any fancy optimization algorithms and simply compute the sum of distances for every possible ranking. This isn’t a bad idea at all if the number of items being ranked is relatively small. Here, since there are five items being ranked, the number of possible rankings is 5! = 120, which is not too big for a modern computer to handle. It may take some time for the exhaustive search approach to yield and answer, but the answer produced by exhaustive search comes with the reassurance that it does, in fact, minimize the sum of distances.

This is in fact what I did for estimating the central ranking when minimizing the sum of Kendall distances from said ranking. The resulting ranking, again, was Seeker/Guardian/Mystic/Rogue/Survivor (which agrees with what we determined just by looking at the pairs matrix; this likely is not a coincidence).

**Statistical Inference**

All of the above I consider falling into the category of descriptive statistics. It describes aspects of the sample without attempting to extrapolate to the rest of the population. With statistical inference we want to see what we can say about the population as a whole.

I should start by saying that the usual assumptions made in statistical inference are likely not satisfied by my sample. It was an opt-in sample; people *chose* to participate. That alone makes it a non-random sample. Additionally, only participants active on Facebook, Reddit, Twitter, Board Game Geek, and the Fantasy Flight forums were targeted by my advertising of the poll. Thus the Arkham players were likely those active on the Internet, likely at a particular time of day and day of the week (given how these websites try to push older content off the main page). They were likely young, male, and engaged enough in the game to be in the community (and unlikely to be a “casual” player). Thus the participants are likely to be more homogenous than the population of Arkham Horror players overall.

Just as a thought experiment, what would be a better study, one where we could feel confident in the inferential ability of our sample? Well, we would grab randomly selected people from the population (perhaps from pulling random names from the phone book), have them join our study, teach them how to play the game, make them play the game for many hours until they could form an educated opinion of the game (probably at least 100 hours), then ask them to rate the classes. This would be high-quality data and we could believe the data is reliable, but *damn* would it be expensive! No one at FFG would consider data of that quality worth the price, and frankly neither would I.

Having said that, while the sample I have is certainly flawed in how it was collected, I actually believe we can get good results from it. The opinions of the participants are likely educated ones, so we probably still have a good idea how the Arkham Horror classes compare to one another.

In rank data analysis there is a probability model called the *uniform distribution* that serves as a starting point for inference. Under the uniform distribution, every ranking vector is equally likely to be observed; in short, there is no preference among the judges among the choices. The marginals matrix should have all entries be \frac{1}{m}, all off-diagonal entries of the pairs matrix should be \frac{1}{2}, and any “central” ranking is meaningless since every ranking is equally likely to be seen. According to the uniform distribution, P(\{\omega\}) = \frac{1}{m!}. If we cannot distinguish our data from data drawn from the uniform distribution, our work is done; we basically say there is no “common” ranking scheme and go about our day.

There are many tests for checking for the uniform distribution, and they are often based on the statistics we’ve already seen, such as the mean rank vector, the marginals matrix, and the pairs matrix. If mis small enough relative to the sample size, we could even just base a test off of how frequently each particular ranking was seen. A test based off the latter could detect any form of non-uniformity in the data, while tests based off the marginals or pairs matrices or the mean vector cannot detect all forms of non-uniformity; that said, they often require much less data to be performed.

As mentioned, I like working with the pairs matrix/Kendall distance. The statistical test, though, involves a vector \hat{\kappa}, which is the aforementioned upper triangle of the pairs matrix (excluding the diagonal entries which are always zero). (More specifically, \hat{\kappa})is a vector containing the upper-diagonal entries of the pairs matrix laid out in [row-major form](https://en.wikipedia.org/wiki/Row-_and_column-major_order).)

The test decides between

H_0: \text{Data was drawn from uniform distribution}

H_A: H_0 \text{ is false}

The test statistic is

12n(\|\hat{\kappa} - \frac{1}{2} 1_{\bar{m}}\|^2 - \|\bar{y} - \frac{m + 1}{2} 1_m\|^2 / (m + 1))

If the null hypothesis is true, then the test statistic, for large n, a \chi^2distribution with \bar{m}degrees of freedom. (For the Arkham Horror classes case, \bar{m} = 10.) Large test statistics are evidence against the null hypothesis, so p-values are the area underneath the \chi^2_{\bar{m}}curve to the right of the test statistic.

For our data set, the reported test statistic was 2309938376; not shockingly, the corresponding p-value is near zero. So the data was not drawn from the uniform distribution. Arkham Horror players do have class preferences.

But what are plausible preferences players could have? We can answer this using a confidence interval. Specifically, we want to know what *rankings* are plausible, and thus what we want is a confidence set of rankings.

Finding a formula for a confidence set of the central ranking is extremely hard to do, but it’s not as hard to form one for one of the statistics we can compute from the rankings, then use the possible values of that statistic to find corresponding plausible central rankings. For example, once could find a confidence set for the mean ranking vector, then translate those mean rankings into ranking vectors (this is what Marden did in his book).

As I said before, I like the pairs matrix/Kendall distance in the rank data context, so I want to form a confidence set for \kappa, the population equivalent of \hat{\kappa}, the key entries of the pairs matrix. To do this, we cannot view the rank data the same way we did before; instead of seeing the m-dimensional vector (2, 1, 4, 3, 5), we need to see the equivalent \bar{m}-dimensional vector (0, 1, 1, 1, 1, 1, 1, 0, 1, 1)that consists only of ones and zeros and records the pair-wise relationships among the ranks, rather than the ranks themselves (the latter vector literally says that item one is not ranked higher than item two, item one is ranked higher than item three, same for four, same for five, then that item two is ranked higher than item three, same for four, same for five, and so on, finally saying in its last entry that item four is ranked higher than item five).

We first compute \hat{\kappa}by taking the means of these vectors. Then we compute the sample [covariance matrix](https://en.wikipedia.org/wiki/Covariance_matrix) of the vectors; call it \hat{\Sigma}. Then a 100(1 - \alpha)% confidence set for the true \kappa, appropriate for large sample sizes, is:

\{\kappa: n(\hat{\kappa} - \kappa)^{T} \hat{\Sigma}^{-1} (\hat{\kappa} - \kappa) \leq \chi^2_{\bar{m}, 1 - \alpha}\}

where \chi^2_{k, 1 - \alpha}is the (1 - \alpha)^{th}percentile of the \chi^2distribution with kdegrees of freedom.

The region I’ve just described is a \bar{m}-dimensional [ellipsoid](https://en.wikipedia.org/wiki/Ellipsoid#In_higher_dimensions), a football-like shape that lives in a space with (probably) more than three dimensions. It sounds daunting, but one can still figure out what rankings are plausible once this region is computed. The trick is to work with each of the coordinates of the vector \kappaand determine whether there is a \kappain the ellipsoid where that coordinate is 1/2. If the answer is no, then the value of that coordinate, for all \kappain the ellipsoid, is either always above or always below 1/2. You can then look to \hat{\kappa}(which is in the dead center of the ellipsoid) to determine which is the case.

What’s the significance of this? Let’s say that you listed all possible rankings in a table. Let’s suppose you did this procedure for the coordinate of \kappacorresponding to the Seeker/Rogue pair. If you determine that this coordinate is not 1/2 and that all \kappain the ellipsoid ranks Seekers above Rogues, then you would take your list of rankings and remove all rankings that Rogues before Seekers, since these rankings are not in the confidence set.

If you do find a $\latex \kappa$ in the ellipsoid where the selected coordinate is 1/2, then you would not eliminate any rows in your list of rankings since you know that your confidence set must include some rankings that rank the two items one way and some rankings where the items are ranked the opposite way.

Repeat this procedure with every coordinate of \kappa—that is, every possible pairing of choices—and you then have a confidence set for central rankings.

Determining whether there is a \kappavector in the ellipsoid with a select coordinate valued at 1/2 can be done via optimization. That is, find a $\latex \kappa$ that minimizes n(\hat{\kappa} - \kappa)^{T} \hat{ \Sigma}^{-1} (\hat{\kappa} - \kappa)subject to the constraint that \kappa_j = 1/2. You don’t even need fancy minimization algorithms for doing this; the minimum can, in principle, be computed analytically with multivariate calculus. After you found a minimizing \kappa, determine what the value of n(\hat{\kappa} - \kappa)^{T} \hat{\Sigma}^{-1} (\hat{\kappa} - \kappa)is at that \kappa. If it is less than \chi^2_{\bar{m}, 1 - \alpha}, then you found a \kappain the ellipsoid; otherwise, you know there is no such \kappa.

This was the procedure I used on the Arkham Horror class ranking data. The 95% confidence interval so computed determined that Seekers were ranked higher than Rogues and Survivors. That means that Seekers cannot have a ranking worse than 3 and Rogues and Survivors could not have rankings better than 2. Any ranking consistent with these constraints, though, is a plausible population central ranking. In fact, this procedure suggested that all the rankings below are plausible central population rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 2 4 3 5

2 1 2 5 3 4

3 1 3 4 2 5

4 1 3 5 2 4

5 1 4 3 2 5

6 1 4 5 2 3

7 1 5 3 2 4

8 1 5 4 2 3

9 2 1 4 3 5

10 2 1 5 3 4

11 2 3 4 1 5

12 2 3 5 1 4

13 2 4 3 1 5

14 2 4 5 1 3

15 2 5 3 1 4

16 2 5 4 1 3

17 3 1 4 2 5

18 3 1 5 2 4

19 3 2 4 1 5

20 3 2 5 1 4

21 3 4 2 1 5

22 3 4 5 1 2

23 3 5 2 1 4

24 3 5 4 1 2

25 4 1 3 2 5

26 4 1 5 2 3

27 4 2 3 1 5

28 4 2 5 1 3

29 4 3 2 1 5

30 4 3 5 1 2

31 4 5 2 1 3

32 4 5 3 1 2

33 5 1 3 2 4

34 5 1 4 2 3

35 5 2 3 1 4

36 5 2 4 1 3

37 5 3 2 1 4

38 5 3 4 1 2

39 5 4 2 1 3

40 5 4 3 1 2

The confidence interval, by design, is much less bold than just an estimate of the most central ranking. Our interval suggests that there’s a lot we don’t know about what the central ranking is; we only know that whatever it is, it ranks Seekers above Rogues and Survivors.

The confidence set here is at least conservative in that it could perhaps contain too many candidate central rankings. I don’t know for sure whether we could improve on the set and eliminate more ranks from the plausible set by querying more from the confidence set for \kappa. Perhaps there are certain combinations that cannot exist, like excluding rankings that give both Seekers and Guardians a high ranking at the same time. If I were a betting man, though, I’d bet that the confidence set found with this procedure could be improved, in that not every vector in the resulting set corresponds with a \kappain the original ellipsoidal confidence set. Improving this set, though, would take a lot of work as one would have to consider multiple coordinates of potential \kappasimultaneously, then find a rule for eliminating ranking vectors based on the results.

**Clustering**

[Matt Newman, the lead designer of Arkham Horror: The Card Game, does not believe all players are the same](https://www.fantasyflightgames.com/en/news/2018/7/13/arkhamtypes/). Specifically, he believes that there are player types that determine how they like to play. In statistics we might say that Matt Newman believes that there are clusters of players within any sufficiently large and well-selected sample of players. This suggests we may want to perform cluster analysis to find these sub-populations.

If you haven’t heard the term before, clustering is the practice of finding “similar” data points, grouping them together, and identifying them as belonging to some sub-population for which no label was directly observed. It’s not unreasonable to believe that these sub-populations exist and so I sought to do clustering myself.

There are many ways to cluster. Prof. Malden said that a clustering of rank data into Lclusters should minimize the sum of the distances of each observation from their assigned cluster’s centers. However, he did not suggest a good algorithm for finding these clusters. He did suggest that for small samples, small mand for a small number of clusters, we could exhaustively search for optimal clusters, an impractical idea.

I initially attempted a [k-means](https://en.wikipedia.org/wiki/K-means_clustering)-type algorithm for finding good clusters, one that used the Kendall distance rather than the Euclidean distance, but unfortunately I could not get the algorithm to give good results. I don’t know whether I have errors in my code (listed below) or whether the algorithm just doesn’t work for Kendall distances, but it didn’t work; in fact, it would take a good clustering and make it worse! I eventually abandoned my home-brewed k-centers algorithm (and the hours of work that went into it) and just used [spectral clustering](https://en.wikipedia.org/wiki/Spectral_clustering).

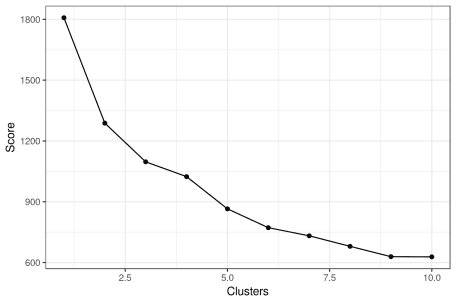
Spectral clustering isn’t easily described, but the idea of spectral clustering is to find groups of data that a random walker, walking from point to point along a weighted graph, would spend a long time in before moving to another group. (That’s the best simplification I can make; the rest is linear algebra.) In order to do spectral clustering, one must have a notion of “similarity” of data points. “[Similarity](https://en.wikipedia.org/wiki/Similarity_measure)” roughly means the opposite of “distance”; in fact, if you have a distance metric (and we do here), you can find a similarity measure by subtracting all distances from the maximum distance between any two objects. Similarity measures are not as strictly defined as distance metrics; any function that gives two “similar” items a high score and two “dissimilar” items a low score could be considered a similarity function.

Spectral clustering takes a matrix of similarity measures, computed for each pair of observations, and spits out cluster assignments. But in addition to the similarity measure, we need to decide how many clusters to find.

I find determining the “best” number of clusters to find the hardest part of clustering. We could have only one cluster, containing all our data; this is what we start with. We could also assign each data point to its own cluster; our aforementioned measure of cluster quality would then be zero, which would be great if it weren’t for the fact that our clusters mean nothing!

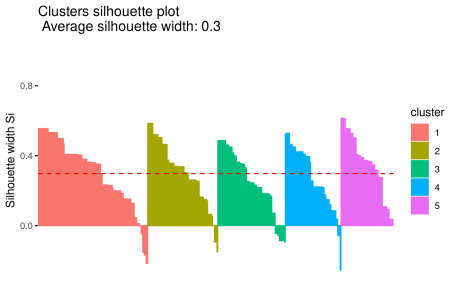
One approach people use for determining how many clusters to pick is the so-called [elbow method](https://en.wikipedia.org/wiki/Elbow_method_(clustering)). You take a plot of, say, Malden’s metric, compared against the number of clusters, and see if you can spot the “elbow” in the plot. The elbow corresponds to the “best” number of clusters.

Here’s the corresponding plot for the dataset here:



If you’re unsure where the “elbow” of the plot is, that’s okay; I’m not sure either. My best guess is that it’s at five clusters; hence my choice of five clusters.

Another plot that people use is the [silhouette plot](https://en.wikipedia.org/wiki/Silhouette_(clustering)), explained quite well by the [scikit-learn documentation](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html). The silhouette plot for the clustering found by spectral clustering is shown below:



Is this a good silhouette plot? I’m not sure. It’s not the worst silhouette plot I saw for this data set but it’s not as good as examples shown in the **scikit-learn** documentation. There are observations that appear to be in the wrong cluster according to the silhouette analysis. So… inconclusive?

I also computed the [Dunn index](https://en.wikipedia.org/wiki/Dunn_index) of the clusters. I never got a value greater than 0.125. All together, these methods lead me to suspect that there are no meaningful clusters in this data set, at least none that can be found with this approach.

But people like cluster analysis, so if you’re one of those folks, I have results for you.

CLUSTERING

----------

Counts: Cluster

1 2 3 4 5

130 83 80 66 62

Centers:

Guardian Mystic Rogue Seeker Survivor

1 3 2 4 1 5

2 3 5 4 1 2

3 3 4 1 2 5

4 1 5 3 4 2

5 5 1 4 3 2

Score: 881

CLUSTER CONFIDENCE INTERVALS

----------------------------

Cluster 1:

With 95% confidence:

Guardian is better than Rogue

Guardian is better than Survivor

Mystic is better than Rogue

Mystic is better than Survivor

Seeker is better than Rogue

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 2 4 3 5

2 1 2 5 3 4

3 1 3 4 2 5

4 1 3 5 2 4

5 2 1 4 3 5

6 2 1 5 3 4

7 2 3 4 1 5

8 2 3 5 1 4

9 3 1 4 2 5

10 3 1 5 2 4

11 3 2 4 1 5

12 3 2 5 1 4

Cluster 2:

With 95% confidence:

Guardian is better than Mystic

Guardian is better than Rogue

Seeker is better than Guardian

Seeker is better than Mystic

Survivor is better than Mystic

Seeker is better than Rogue

Survivor is better than Rogue

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 2 4 5 1 3

2 2 5 4 1 3

3 3 4 5 1 2

4 3 5 4 1 2

Cluster 3:

With 95% confidence:

Rogue is better than Guardian

Rogue is better than Mystic

Rogue is better than Seeker

Rogue is better than Survivor

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 2 3 1 4 5

2 2 4 1 3 5

3 2 5 1 3 4

4 3 2 1 4 5

5 3 4 1 2 5

6 3 5 1 2 4

7 4 2 1 3 5

8 4 3 1 2 5

9 4 5 1 2 3

10 5 2 1 3 4

11 5 3 1 2 4

12 5 4 1 2 3

Cluster 4:

With 95% confidence:

Guardian is better than Mystic

Guardian is better than Seeker

Rogue is better than Mystic

Survivor is better than Mystic

Survivor is better than Seeker

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 4 2 5 3

2 1 4 3 5 2

3 1 5 2 4 3

4 1 5 3 4 2

5 1 5 4 3 2

6 2 4 1 5 3

7 2 4 3 5 1

8 2 5 1 4 3

9 2 5 3 4 1

10 2 5 4 3 1

11 3 4 1 5 2

12 3 4 2 5 1

13 3 5 1 4 2

14 3 5 2 4 1

Cluster 5:

With 95% confidence:

Mystic is better than Guardian

Survivor is better than Guardian

Mystic is better than Rogue

Mystic is better than Seeker

Survivor is better than Rogue

Survivor is better than Seeker

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 3 1 4 5 2

2 3 1 5 4 2

3 3 2 4 5 1

4 3 2 5 4 1

5 4 1 3 5 2

6 4 1 5 3 2

7 4 2 3 5 1

8 4 2 5 3 1

9 5 1 3 4 2

10 5 1 4 3 2

11 5 2 3 4 1

12 5 2 4 3 1

When computing confidence sets for clusters I ran into an interesting problem: what if, say, you never see Seekers ranked below Guardians? This will cause one of the entries of \hat{\kappa}to be either 0 or 1, and there is no “variance” in its value; it’s always the same. This will cause the covariance matrix to be non-invertible since it has rows/columns that are zero. The solution to this is to eliminate those rows and work only with the non-constant entries of \hat{\kappa}. That said, I still treat the entries removed as if they were “statisticall significant” results and remove rankings from our confidence set that are inconsistent with what we saw in the data. In short, if Seekers are never ranked below Guardians, remove all rankings in the confidence set that rank Seekers below Guardians.

One usually isn’t satisfied with just a clustering; it would be nice to determine what a clustering signifies about those who are in the cluster. For instance, what type of player gets assigned to Cluster 1? I feel that inspecting the data in a more thoughtful and manual way can give a sense to what characteristic individuals assigned to a cluster share. For instance, I read the comments submitted by poll participants to hypothesize what types of players were being assigned to particular clusters. You can read these comments at the bottom of this article, after the code section.

**Code**

All source code used to do the rank analysis done here is listed below, in a .R file intended to be run as an executable from a command line. (I created and ran it on a Linux system.)

Several packages had useful functions specific for this type of analysis, such as **pmr** (meant for modelling rand data) and **rankdist** (which had a lot of tools for working with the Kendall distance). The confidence interval, central ranking estimator, and hypothesis testing tools, though, I wrote myself, and they may not exist elsewhere.

I at least feel that the script itself is well-documented and I no longer need to explain it. But I will warn others that it was tailored to my problem, and the methods employed may not work well with larger sample sizes or when more items need to be ranked.

**Conclusion**

This is only the tip of the iceberg for rank data analysis. We have not even touched on modelling for rank data, which can provide even richer inference. If you’re interested, I’ll refer you again to Malden’s book.

I enjoyed this analysis so much I [asked a Reddit question](https://www.reddit.com/r/statistics/comments/ap89he/is_it_possible_for_a_poor_curious_college_student/) about where else I could conduct surveys (while at the same time still being statistically sound) because I’d love to do it again. I feel like there’s much to learn from rank data; it has great potential. Hopefully this article sparked your interest too.

**R Script for Analysis**

#!/usr/bin/Rscript

################################################################################

# ArkhamHorrorClassPreferenceAnalysis.R

################################################################################

# 2019-02-10

# Curtis Miller

################################################################################

# Analyze Arkham Horror LCG class preference survey data.

################################################################################

# optparse: A package for handling command line arguments

if (!suppressPackageStartupMessages(require("optparse"))) {

install.packages("optparse")

require("optparse")

}

################################################################################

# CONSTANTS

################################################################################

CLASS\_COUNT <- 5

CLASSES <- c("Guardian", "Mystic", "Rogue", "Seeker", "Survivor")

CLASS\_COLORS <- c("Guardian" = "#00628C",

"Mystic" = "#44397D",

"Rogue" = "#17623B",

"Seeker" = "#B87D37",

"Survivor" = "#AA242D")

################################################################################

# FUNCTIONS

################################################################################

`%s%` <- function(x, y) {paste(x, y)}

`%s0%` <- function(x, y) {paste0(x, y)}

#' Sum of Kendall Distances

#'

#' Given a ranking vector and a matrix of rankings, compute the sum of Kendall

#' distances.

#'

#' @param r The ranking vector

#' @param mat The matrix of rankings, with each row having its own ranking

#' @param weight Optional vector weighting each row of \code{mat} in the sum,

#' perhaps representing how many times that ranking is repeated

#' @return The (weighted) sum of the Kendall distances

#' @examples

#' mat <- rbind(1:3,

#' 3:1)

#' skd(c(2, 1, 3), mat)

skd <- function(r, mat, weight = 1) {

dr <- partial(DistancePair, r2 = r)

sum(apply(mat, 1, dr) \* weight)

}

#' Least Sum of Kendall Distances Estimator

#'

#' Estimates the "central" ranking by minimizing the sum of Kendall distances,

#' via exhaustive search.

#'

#' @param mat The matrix of rankings, with each row having its own ranking

#' @param weight Optional vector weighting each row of \code{mat} in the sum,

#' perhaps representing how many times that ranking is repeated

#' @return Ranking vector that minimizes the (weighted) sum of rankings

#' @examples

#' mat <- rbind(1:3,

#' 3:1)

#' lskd\_estimator(mat)

lskd\_estimator <- function(mat, weight = NULL) {

if (is.null(weight)) {

reduced <- rank\_vec\_count(mat)

mat <- reduced$mat

weight <- reduced$count

}

skdm <- partial(skd, mat = mat, weight = weight)

m <- max(mat)

permutation\_mat <- permutations(m, m)

sums <- apply(permutation\_mat, 1, skdm)

permutation\_mat[which.min(sums),]

}

#' Identify Ranking With Center

#'

#' Find the index of the center closest to a ranking vector.

#'

#' @param r The ranking vector

#' @param mat The matrix of rankings, with each row having its own ranking

#' @return Index of row that is closest to \code{r}

#' @examples

#' mat <- rbind(1:3,

#' 3:1)

#' close\_center(c(2, 1, 3), mat)

close\_center <- function(r, mat) {

dr <- partial(DistancePair, r2 = r)

which.min(apply(mat, 1, dr))

}

#' Simplify Rank Matrix To Unique Rows

#'

#' Given a matrix with rows representing rankings, this function reduced the

#' matrix to rows of only unique rankings and also counts how many times a

#' ranking appeared.

#'

#' @param mat The matrix of rankings, with each row having its own ranking

#' @return A list with entries \code{"mat"} and \code{"count"}, with

#' \code{"mat"} being a matrix now with unique rankings and

#' \code{"count"} being a vector of times each row in new matrix

#' appeared in the old matrix

#' @examples

#' mat <- rbind(1:3,

#' 3:1)

#' rank\_vec\_count(mat)

rank\_vec\_count <- function(mat) {

old\_col\_names <- colnames(mat)

old\_row\_names <- rownames(mat)

res\_df <- aggregate(list(numdup = rep(1, times = nrow(mat))),

as.data.frame(mat), length)

count <- res\_df$numdup

new\_mat <- res\_df[1:ncol(mat)]

colnames(new\_mat) <- old\_col\_names

rownames(new\_mat) <- old\_row\_names

list("mat" = as.matrix(new\_mat), "count" = count)

}

#' Find \eqn{k} Ranking Clusters

#'

#' Estimate \eqn{k} clusters of rankings.

#'

#' The algorithm to find the ranking clusters resembles the \eqn{k}-means++

#' algorithm except that the distance metric is the Kendall distance.

#'

#' @param mat The matrix of rankings, with each row having its own ranking

#' @param k The number of clusters to find

#' @param max\_iter The maximum number of iterations for algorithm

#' @param tol The numerical tolerance at which to end the algorithm if met

#' @return A list containing the central rankings of each cluster (in

#' \code{"centers"}) and a vector with integers representing cluster

#' assignments

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' rank\_cluster(mat, 2)

rank\_cluster <- function(mat, k, init\_type = c("spectral", "kmeans++"),

max\_iter = 100, tol = 1e-4) {

simplified\_mat <- rank\_vec\_count(mat)

mat <- simplified\_mat$mat

count <- simplified\_mat$count

init\_type <- init\_type[1]

if (init\_type == "kmeans++") {

centers <- rank\_cluster\_center\_init(mat, k)

} else if (init\_type == "spectral") {

centers <- rank\_cluster\_spectral(mat, k)$centers

} else {

stop("Don't know init\_type" %s% init\_type)

}

old\_centers <- centers

cc\_centers <- partial(close\_center, mat = centers)

clusters <- apply(mat, 1, cc\_centers)

for (iter in 1:max\_iter) {

centers <- find\_cluster\_centers(mat, clusters, count)

stopifnot(all(dim(centers) == dim(old\_centers)))

cc\_centers <- partial(close\_center, mat = centers)

clusters <- apply(mat, 1, cc\_centers)

if (center\_distance\_change(centers, old\_centers) < tol) {

break

} else {

old\_centers <- centers

}

}

if (iter == max\_iter) {warning("Maximum iterations reached")}

colnames(centers) <- colnames(mat)

list("centers" = centers, "clusters" = rep(clusters, times = count))

}

#' Find the Distance Between Two Ranking Matrices

#'

#' Find the distance between two ranking matrices by summing the distance

#' between each row of the respective matrices.

#'

#' @param mat1 First matrix of ranks

#' @param mat2 Second matrix of ranks

#' @return The sum of distances between rows of \code{mat1} and \code{mat2}

#' @examples

#' mat <- rbind(1:3,

#' 3:1)

#' center\_distance\_change(mat, mat)

center\_distance\_change <- function(mat1, mat2) {

if (any(dim(mat1) != dim(mat2))) {stop("Dimensions of matrices don't match")}

sum(sapply(1:nrow(mat1), function(i) {DistancePair(mat1[i, ], mat2[i, ])}))

}

#' Initialize Cluster Centers

#'

#' Find initial cluster centers as prescribed by the \eqn{k}-means++ algorithm.

#'

#' @param mat The matrix of rankings, with each row having its own ranking

#' @param k The number of clusters to find

#' @return A matrix containing cluster centers.

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' rank\_cluster\_center\_init(mat, 2)

rank\_cluster\_center\_init <- function(mat, k) {

n <- nrow(mat)

center <- mat[sample(1:n, 1), ]

centers\_mat <- rbind(center)

for (i in 2:k) {

min\_distances <- sapply(1:n, function(l) {

min(sapply(1:(i - 1), function(j) {

DistancePair(mat[l, ], centers\_mat[j, ])

}))

})

center <- mat[sample(1:n, 1, prob = min\_distances/sum(min\_distances)), ]

centers\_mat <- rbind(centers\_mat, center)

}

rownames(centers\_mat) <- NULL

colnames(centers\_mat) <- colnames(mat)

centers\_mat

}

#' Evaluation Metric for Clustering Quality

#'

#' Evaluates a clustering's quality by summing the distance of each observation

#' to its assigned cluster center.

#'

#' @param mat Matrix of rankings (in the rows); the data

#' @param centers Matrix of rankings (in the rows) representing the centers of

#' the clusters

#' @param clusters Vector of indices corresponding to cluster assignments (the

#' rows of the \code{clusters} matrix)

#' @return Score of the clustering

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' centers <- rbind(1:3, 3:1)

#' clusters <- c(1, 1, 2, 2)

#' clustering\_score(mat, centers, clusters)

clustering\_score <- function(mat, centers, clusters) {

sum(sapply(1:nrow(centers), function(i) {

center <- centers[i, ]

submat <- mat[which(clusters == i), ]

skd(center, submat)

}))

}

#' Clustering with Restarts

#'

#' Clusters multiple times and returns the clustering with the lowest clustering

#' score

#'

#' @param ... Parameters to pass to \code{\link{rank\_cluster}}

#' @param restarts Number of restarts

#' @return A list containing the central rankings of each cluster (in

#' \code{"centers"}) and a vector with integers representing cluster

#' assignments

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' rank\_cluster\_restarts(mat, 2, 5)

rank\_cluster\_restarts <- function(mat, ..., restarts = 10) {

best\_score <- Inf

rank\_cluster\_args <- list(...)

rank\_cluster\_args$mat <- mat

for (i in 1:restarts) {

new\_cluster\_scheme <- do.call(rank\_cluster, rank\_cluster\_args)

score <- clustering\_score(mat, new\_cluster\_scheme$centers,

new\_cluster\_scheme$clusters)

if (score < best\_score) {

best\_score <- score

best\_scheme <- new\_cluster\_scheme

}

}

return(best\_scheme)

}

#' Given Clusters, Find Centers

#'

#' Given a collection of clusters, find centers for the clusters.

#'

#' @param mat Matrix of rankings (in rows)

#' @param clusters Vector containing integers identifying cluster assignments,

#' where the integers range from one to the number of clusters

#' @param weight Optional vector weighting each row of \code{mat} in the sum,

#' perhaps representing how many times that ranking is repeated

#' @return Ranking vector that minimizes the (weighted) sum of rankings

#' @return A matrix of ranks representing cluster centers

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' find\_cluster\_centers(mat, c(1, 1, 2, 2))

find\_cluster\_centers <- function(mat, clusters, weight = NULL) {

if (is.null(weight)) {

weight <- rep(1, times = nrow(mat))

}

centers <- t(sapply(unique(clusters), function(i) {

submat <- mat[which(clusters == i), ]

subweight <- weight[which(clusters == i)]

lskd\_estimator(submat, subweight)

}))

colnames(centers) <- colnames(mat)

centers

}

#' Cluster Rankings Via Spectral Clustering

#'

#' Obtain a clustering of rank data via spectral clustering.

#'

#' @param mat Matrix containing rank data

#' @param k Number of clusters to find

#' @return A list with entries: \code{"centers"}, the centers of the clusters;

#' and \code{"clusters"}, a vector assigning rows to clusters.

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' rank\_cluster\_spectral(mat, 2)

rank\_cluster\_spectral <- function(mat, k = 2) {

dist\_mat <- DistanceMatrix(mat)

sim\_mat <- max(dist\_mat) - dist\_mat

clusters <- spectralClustering(sim\_mat, k)

centers <- find\_cluster\_centers(mat, clusters)

list("centers" = centers, "clusters" = clusters)

}

#' Compute the Test Statistic for Uniformity Based on the Pairs Matrix

#'

#' Compute a test for uniformity based on the estimated pairs matrix.

#'

#' Let \eqn{m} be the number of items ranked and \eqn{n} the size of the data

#' set. Let \eqn{\bar{k} = k(k - 1)/2} and \eqn{\bar{y}} the mean rank vector.

#' Let \eqn{\hat{K}^\*} be the upper-triangular part of the estimated pairs

#' matrix (excluding the diagonal), laid out as a vector in row-major order.

#' Finally, let \eqn{1\_k} be a vector of \eqn{k} ones. Then the test statistic

#' is

#'

#' \deqn{12n(\|\hat{K}^\* - \frac{1}{2} 1\_{\bar{m}}\|^2 - \|\bar{y} - \frac{m +

#' 1}{2} 1\_m\|^2 / (m + 1))}

#'

#' Under the null hypothesis this statistic asympotically follow a \eqn{\chi^2}

#' distribution with \eqn{\bar{m}} degrees of freedom.

#'

#' @param mat The data matrix, with rankings in rows

#' @return The value of the test statistic

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' pairs\_uniform\_test\_stat(mat)

pairs\_uniform\_test\_stat <- function(mat) {

desc\_stat <- suppressMessages(destat(mat))

mean\_rank <- desc\_stat$mean.rank

pair <- desc\_stat$pair

m <- ncol(mat) - 1

n <- nrow(mat)

mbar <- choose(m, 2)

K <- pair[upper.tri(pair, diag = FALSE)]

meanK <- rep(1/2, times = mbar)

cm <- rep((m + 1)/2, times = m)

12 \* n \* (sum((K - meanK)^2) - sum((mean\_rank - cm)^2)/(m + 1))

}

#' Compute Covariance Matrix of Pairs Matrix Upper Triangle

#'

#' Compute the covariance matrix of the pairs matrix estimator.

#'

#' @param mat Data matrix, with each ranking having its own row

#' @return The \eqn{m(m - 1)/2}-square matrix representing the covariance matrix

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' pairs\_mat\_cov(mat)

pairs\_mat\_cov <- function(mat) {

n <- nrow(mat)

m <- ncol(mat)

pair <- kappa\_est(mat)

pair <- as.matrix(pair)

# Transform data into a dataset of pair-wise rank comparisons

if (m == 1) {

return(0)

}

kappa\_data <- sapply(2:m, function(j) {mat[, j] > mat[, 1]})

for (i in 2:(m - 1)) {

kappa\_data <- cbind(kappa\_data, sapply((i + 1):m, function(j) {

mat[, j] > mat[, i]

}))

}

kappa\_data <- kappa\_data + 0 # Converts to integers

cov(kappa\_data)

}

#' Estimate \eqn{\kappa} Vector

#'

#' Estimate the \eqn{\kappa} vector, which fully defines the pairs matrix.

#'

#' @param mat Data matrix, with each ranking having its own row

#' @return The \eqn{m(m - 1)/2}-dimensional vector

#' @examples

#' mat <- rbind(1:3,

#' 3:1,

#' c(2, 1, 3),

#' c(3, 1, 2))

#' kappa\_est(mat)

kappa\_est <- function(mat) {

n <- nrow(mat)

df <- as.data.frame(mat)

df$n <- 1

pair <- suppressMessages(destat(df))

pair <- t(pair$pair)

pair <- pair[lower.tri(pair, diag = FALSE)]/n

pair

}

#' Get Plausible Rankings For Central Ranking Based on Kendall Distance

#'

#' Determine a set of plausible central rankings based on the Kendall distance.

#'

#' Let \eqn{\alpha} be one minus the confidence level, \eqn{m} the number of

#' options, \eqn{\bar{m} = m(m - 1)/2}, \eqn{\kappa} the vectorized

#' upper-triangle of the pairs matrix of the population, \eqn{\hat{\kappa}} the

#' sample estimate of \eqn{\kappa}, and \eqn{\hat{\Sigma}} the estimated

#' covariance matrix of \eqn{\hat{kappa}}. Then the approximate \eqn{100(1 -

#' \alpha)}% confidence interval for \eqn{\kappa} is

#'

#' \deqn{\kappa: (\hat{\kappa} - \kappa)^T \hat{\Sigma}^{-1} (\hat{kappa} -

#' \kappa) < \chi^2\_{\bar{m}}}

#'

#' One we have such an interval the next task is to determine which ranking

#' vectors are consistent with plausible \eqn{\kappa}. To do this, the function

#' determines which choices could plausibly be tied according to the confidence

#' interval; that is, which entries of \eqn{\kappa} could plausibly be

#' \eqn{1/2}. Whenever this is rejected, there is a statistically significant

#' difference in the preference of the two choices; looking at \hat{\kappa} can

#' determine which of the two choices is favored. All ranking vectors that would

#' agree that disagree with that preference are eliminated from the space of

#' plausible central ranking vectors. The ranking vectors surviving at the end

#' of this process constitute the confidence interval.

#'

#' @param mat Matrix of rank data, each observation having its own row

#' @param conf\_level Desired confidence level

#' @return A list with entries \code{"ranks"} holding the matrix of plausible

#' rankings in the confidence interval and \code{"preference\_string"}, a

#' string enumerating which options are, with statistical significance,

#' preferred over others

#' @examples

#' mat <- t(replicate(100, {sample(1:3)}))

#' kendall\_rank\_conf\_interval(mat)

kendall\_rank\_conf\_interval <- function(mat, conf\_level = 0.95) {

n <- nrow(mat)

m <- max(mat)

mbar <- choose(m, 2)

kap <- kappa\_est(mat)

Sigma <- pairs\_mat\_cov(mat)

crit\_value <- qchisq(1 - conf\_level, df = mbar, lower.tail = FALSE)

# Find bad rows of Sigma, where the covariance is zero; that variable must be

# constant

const\_vars <- which(colSums(Sigma^2) == 0)

safe\_vars <- which(colSums(Sigma^2) > 0)

safe\_kap <- kap[safe\_vars]

safe\_Sigma <- Sigma[safe\_vars, safe\_vars]

# Determine if hyperplanes where one coordinate is 1/2 intersect confidence

# set

b <- as.matrix(solve(safe\_Sigma, safe\_kap))

a <- t(safe\_kap) %\*% b

a <- a[1, 1]

check\_half <- partial(hei\_check, x = 1/2, A = safe\_Sigma, b = -2 \* b,

d = crit\_value/n - a, invert\_A = TRUE)

sig\_diff\_safe\_vars <- !sapply(1:length(safe\_vars), check\_half)

if (length(const\_vars) > 0) {

sig\_diff <- rep(NA, times = mbar)

sig\_diff[safe\_vars] <- sig\_diff\_safe\_vars

sig\_diff[const\_vars] <- TRUE

} else {

sig\_diff <- sig\_diff\_safe\_vars

}

idx\_matrix <- matrix(0, nrow = m, ncol = m)

idx\_matrix[lower.tri(idx\_matrix, diag = FALSE)] <- 1:mbar

idx\_matrix <- t(idx\_matrix)

rownames(idx\_matrix) <- colnames(mat)

colnames(idx\_matrix) <- colnames(mat)

# Remove rows of potential centers matrix to reflect confidence interval

# results; also, record which groups seem to have significant difference in

# ranking

rank\_string <- ""

permutation\_mat <- permutations(m, m)

for (i in 1:(m - 1)) {

for (j in (i + 1):m) {

sig\_diff\_index <- idx\_matrix[i, j]

if (sig\_diff[sig\_diff\_index]) {

direction <- sign(kap[sig\_diff\_index] - 1/2)

if (direction > 0) {

# Row option (i) is preferred to column option (j)

permutation\_mat <- permutation\_mat[permutation\_mat[, i] <

permutation\_mat[, j], ]

rank\_string <- rank\_string %s0% colnames(mat)[i] %s%

"is better than" %s% colnames(mat)[j] %s0% '\n'

} else if (direction < 0) {

# Row option (i) is inferior to column option (j)

permutation\_mat <- permutation\_mat[permutation\_mat[, i] >

permutation\_mat[, j], ]

rank\_string <- rank\_string %s0% colnames(mat)[j] %s%

"is better than" %s% colnames(mat)[i] %s0% '\n'

}

}

}

}

colnames(permutation\_mat) <- colnames(mat)

return(list("ranks" = permutation\_mat, "preference\_string" = rank\_string))

}

#' Straight Hyperplane and Ellipse Intersection Test

#'

#' Test whether a hyperplane parallel to an axis intersects an ellipse.

#'

#' The ellipse is fully determined by the parameters \code{A}, \code{b}, and

#' \code{d}; in fact, the ellipse consists of all \eqn{x} such that

#'

#' \deqn{x^T A x + b^T x \leq d}

#'

#' \code{x} is the intercept of the hyperplane and \code{k} is the coordinate

#' that is fixed to the value \code{x} and thus determine along which axis the

#' hyperplane is parallel. A value of \code{TRUE} means that there is an

#' intersection, while \code{FALSE} means there is no intersection.

#'

#' @param x The fixed value of the hyperplane

#' @param k The coordinate fixed to \code{x}

#' @param A A \eqn{n \times n} matrix

#' @param b An \eqn{n}-dimensional vector

#' @param d A scalar representing the upper bound of the ellipse

#' @return \code{TRUE} or \code{FALSE} depending on whether the hyperplane

#' intersects the ellipse or not

#' @examples

#' hei\_check(1, 2, diag(3), rep(0, times = 3), 10)

hei\_check <- function(x, k, A, b, d, invert\_A = FALSE) {

b <- as.matrix(b)

n <- nrow(b)

stopifnot(k >= 1 & k <= n)

stopifnot(nrow(A) == ncol(A) & nrow(A) == n)

stopifnot(all(eigen(A)$values > 0))

all\_but\_k <- (1:n)[which(1:n != k)]

s <- rep(0, times = n)

s[k] <- x

s <- as.matrix(s)

if (invert\_A) {

tb <- as.matrix(solve(A, s))

} else {

tb <- A %\*% s

}

td <- t(s) %\*% tb + t(b) %\*% s

if (invert\_A) {

# XXX: curtis: NUMERICALLY BAD; FIX THIS -- Thu 14 Feb 2019 07:50:19 PM MST

A <- solve(A)

}

tA <- A[all\_but\_k, all\_but\_k]

tx <- -solve(tA, (b/2 + tb)[all\_but\_k, ])

tx <- as.matrix(tx)

val <- t(tx)%\*% tA %\*% tx + t((b + 2 \* tb)[all\_but\_k]) %\*% tx + td - d

val <- val[1, 1]

val <= 0

}

################################################################################

# MAIN FUNCTION DEFINITION

################################################################################

main <- function(input, prefix = "", width = 6, height = 4, clusters = 5,

conflevel = 95, comments = "AHLCGClusterComments.txt",

detailed = FALSE, help = FALSE) {

suppressPackageStartupMessages(library(pmr))

suppressPackageStartupMessages(library(ggplot2))

suppressPackageStartupMessages(library(reshape2))

suppressPackageStartupMessages(library(dplyr))

suppressPackageStartupMessages(library(rankdist))

suppressPackageStartupMessages(library(gtools))

suppressPackageStartupMessages(library(purrr))

suppressPackageStartupMessages(library(anocva))

load(input)

n <- nrow(survey\_data)

rank\_data <- survey\_data[CLASSES]

rank\_data$n <- 1

rank\_mat <- as.matrix(survey\_data[CLASSES])

# Get basic descriptive statistics: mean ranks, marginals, pairs

desc\_stat <- suppressMessages(destat(rank\_data))

mean\_rank <- desc\_stat$mean.rank

marginal <- desc\_stat$mar

pair <- desc\_stat$pair

names(mean\_rank) <- CLASSES

rownames(marginal) <- CLASSES

colnames(marginal) <- 1:CLASS\_COUNT

rownames(pair) <- CLASSES

colnames(pair) <- CLASSES

# Compute "typical" distance based on least sum of Kendall distances

best\_rank <- lskd\_estimator(rank\_mat)

names(best\_rank) <- CLASSES

# Hypothesis Testing for Uniformity

statistic <- pairs\_uniform\_test\_stat(rank\_data)

# Confidence Interval

ci <- kendall\_rank\_conf\_interval(rank\_mat, conf\_level = conflevel / 100)

# Cluster data

rank\_clustering <- rank\_cluster\_spectral(rank\_mat, k = clusters)

centers <- rank\_clustering$centers

Cluster <- rank\_clustering$clusters # Naming convention broke for printing

rownames(centers) <- 1:nrow(centers)

# Plotting

marginal\_plot <- ggplot(

melt(100 \* marginal / n, varnames = c("Class", "Rank"),

value.name = "Percent"),

aes(fill = Class, x = Class, y = Percent, group = Rank)) +

geom\_bar(position = "dodge", stat = "identity") +

scale\_fill\_manual(values = CLASS\_COLORS) +

labs(title = "Class Rankings") +

theme\_bw()

ggsave(prefix %s0% "marginal\_plot.png", plot = marginal\_plot,

width = width, height = height, units = "in", dpi = 300)

pair\_plot <- ggplot(

melt(100 \* pair / n, varnames = c("Class", "Opposite"),

value.name = "Percent") %>% filter(Percent > 0),

aes(fill = Opposite, x = Class, y = Percent)) +

geom\_bar(position = "dodge", stat = "identity") +

geom\_hline(yintercept = 50, linetype = 2, color = "red") +

scale\_fill\_manual(values = CLASS\_COLORS) +

labs(title = "Class Ranking Comparison") +

theme\_bw()

ggsave(prefix %s0% "pair\_plot.png", plot = pair\_plot, width = width,

height = height, units = "in", dpi = 300)

# Place cluster comments in file

comment\_string <- ""

for (i in 1:clusters) {

comment\_string <- comment\_string %s0%

"\n\nCLUSTER" %s% i %s0% "\n------------\n\n" %s0%

paste(survey\_data$Reason[survey\_data$Reason != "" & Cluster == i],

collapse = "\n\n-\*-\n\n")

}

cat(comment\_string, file = comments)

# Printing

cat("\nMEAN RANK\n---------\n")

print(round(mean\_rank, digits = 2))

cat("\nMARGINALS\n---------\n")

print(round(100 \* marginal / n, digits = 2))

cat("\nPAIRS\n-----\n")

print(round(100 \* pair / n, digits = 2))

cat("\nUNIFORMITY TEST\n---------------\n")

cat("Test Statistic:", statistic, "\n")

cat("P-value:", pchisq(statistic, df = choose(CLASS\_COUNT, 2),

lower.tail = FALSE), "\n")

cat("\nOPTIMAL RANK ESTIMATE\n---------------------\n")

print(sort(best\_rank))

cat("\nWith", conflevel %s0% '%', "confidence:",

'\n' %s0% ci$preference\_string)

if (detailed) {

cat("\nPlausible Modal Rankings:\n")

print(as.data.frame(ci$ranks))

}

cat("\nCLUSTERING\n----------\nCounts: ")

print(table(Cluster))

cat("\nCenters:\n")

print(centers)

cat("\nScore:", clustering\_score(rank\_mat, centers, Cluster), "\n")

if (detailed) {

cat("\nCLUSTER CONFIDENCE INTERVALS\n----------------------------\n")

for (i in 1:clusters) {

cat("\nCluster", i %s0% ':\n')

ci\_cluster <- kendall\_rank\_conf\_interval(rank\_mat[Cluster == i, ])

cat("\nWith", conflevel %s0% '%', "confidence:",

'\n' %s0% ci\_cluster$preference\_string)

cat("\nPlausible Modal Rankings:\n")

print(as.data.frame(ci\_cluster$ranks))

}

}

}

################################################################################

# INTERFACE SETUP

################################################################################

if (sys.nframe() == 0) {

cl\_args <- parse\_args(OptionParser(

description = paste("Analyze Arkham Horror LCG class preference survey",

"data and print results."),

option\_list = list(

make\_option(c("--input", "-i"), type = "character",

help = paste("Input file containing survey data")),

make\_option(c("--prefix", "-p"), type = "character", default = "",

help = "Another command-line argument"),

make\_option(c("--width", "-w"), type = "double", default = 6,

help = "Width of plots"),

make\_option(c("--height", "-H"), type = "double", default = 4,

help = "Height of plots"),

make\_option(c("--clusters", "-k"), type = "integer", default = 5,

help = "Number of clusters in spectral clustering"),

make\_option(c("--comments", "-c"), type = "character",

default = "AHLCGClusterComments.txt",

help = "File to store participant comments organized" %s%

"by cluster"),

make\_option(c("--conflevel", "-a"), type = "double", default = 95,

help = "Confidence level of confidence set"),

make\_option(c("--detailed", "-d"), action = "store\_true",

default = FALSE, help = "More detail in report")

)

))

do.call(main, cl\_args)

}

**Report**

$ ./ArkhamHorrorClassPreferenceAnalysis.R -i AHLCGClassPreferenceSurveys.Rda --detailed

MEAN RANK

---------

Guardian Mystic Rogue Seeker Survivor

2.92 3.10 3.16 2.60 3.22

MARGINALS

---------

1 2 3 4 5

Guardian 18.29 20.43 26.84 19.71 14.73

Mystic 19.71 18.29 17.81 20.90 23.28

Rogue 19.24 14.73 20.67 21.38 23.99

Seeker 28.03 25.18 17.10 18.53 11.16

Survivor 14.73 21.38 17.58 19.48 26.84

PAIRS

-----

Guardian Mystic Rogue Seeker Survivor

Guardian 0.00 54.16 55.34 42.52 55.82

Mystic 45.84 0.00 51.07 39.90 53.44

Rogue 44.66 48.93 0.00 38.72 51.54

Seeker 57.48 60.10 61.28 0.00 61.52

Survivor 44.18 46.56 48.46 38.48 0.00

UNIFORMITY TEST

---------------

Test Statistic: 2309938376

P-value: 0

OPTIMAL RANK ESTIMATE

---------------------

Seeker Guardian Mystic Rogue Survivor

1 2 3 4 5

With 95% confidence:

Seeker is better than Rogue

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 2 4 3 5

2 1 2 5 3 4

3 1 3 4 2 5

4 1 3 5 2 4

5 1 4 3 2 5

6 1 4 5 2 3

7 1 5 3 2 4

8 1 5 4 2 3

9 2 1 4 3 5

10 2 1 5 3 4

11 2 3 4 1 5

12 2 3 5 1 4

13 2 4 3 1 5

14 2 4 5 1 3

15 2 5 3 1 4

16 2 5 4 1 3

17 3 1 4 2 5

18 3 1 5 2 4

19 3 2 4 1 5

20 3 2 5 1 4

21 3 4 2 1 5

22 3 4 5 1 2

23 3 5 2 1 4

24 3 5 4 1 2

25 4 1 3 2 5

26 4 1 5 2 3

27 4 2 3 1 5

28 4 2 5 1 3

29 4 3 2 1 5

30 4 3 5 1 2

31 4 5 2 1 3

32 4 5 3 1 2

33 5 1 3 2 4

34 5 1 4 2 3

35 5 2 3 1 4

36 5 2 4 1 3

37 5 3 2 1 4

38 5 3 4 1 2

39 5 4 2 1 3

40 5 4 3 1 2

CLUSTERING

----------

Counts: Cluster

1 2 3 4 5

130 83 80 66 62

Centers:

Guardian Mystic Rogue Seeker Survivor

1 3 2 4 1 5

2 3 5 4 1 2

3 3 4 1 2 5

4 1 5 3 4 2

5 5 1 4 3 2

Score: 881

CLUSTER CONFIDENCE INTERVALS

----------------------------

Cluster 1:

With 95% confidence:

Guardian is better than Rogue

Guardian is better than Survivor

Mystic is better than Rogue

Mystic is better than Survivor

Seeker is better than Rogue

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 2 4 3 5

2 1 2 5 3 4

3 1 3 4 2 5

4 1 3 5 2 4

5 2 1 4 3 5

6 2 1 5 3 4

7 2 3 4 1 5

8 2 3 5 1 4

9 3 1 4 2 5

10 3 1 5 2 4

11 3 2 4 1 5

12 3 2 5 1 4

Cluster 2:

With 95% confidence:

Guardian is better than Mystic

Guardian is better than Rogue

Seeker is better than Guardian

Seeker is better than Mystic

Survivor is better than Mystic

Seeker is better than Rogue

Survivor is better than Rogue

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 2 4 5 1 3

2 2 5 4 1 3

3 3 4 5 1 2

4 3 5 4 1 2

Cluster 3:

With 95% confidence:

Rogue is better than Guardian

Rogue is better than Mystic

Rogue is better than Seeker

Rogue is better than Survivor

Seeker is better than Survivor

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 2 3 1 4 5

2 2 4 1 3 5

3 2 5 1 3 4

4 3 2 1 4 5

5 3 4 1 2 5

6 3 5 1 2 4

7 4 2 1 3 5

8 4 3 1 2 5

9 4 5 1 2 3

10 5 2 1 3 4

11 5 3 1 2 4

12 5 4 1 2 3

Cluster 4:

With 95% confidence:

Guardian is better than Mystic

Guardian is better than Seeker

Rogue is better than Mystic

Survivor is better than Mystic

Survivor is better than Seeker

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 1 4 2 5 3

2 1 4 3 5 2

3 1 5 2 4 3

4 1 5 3 4 2

5 1 5 4 3 2

6 2 4 1 5 3

7 2 4 3 5 1

8 2 5 1 4 3

9 2 5 3 4 1

10 2 5 4 3 1

11 3 4 1 5 2

12 3 4 2 5 1

13 3 5 1 4 2

14 3 5 2 4 1

Cluster 5:

With 95% confidence:

Mystic is better than Guardian

Survivor is better than Guardian

Mystic is better than Rogue

Mystic is better than Seeker

Survivor is better than Rogue

Survivor is better than Seeker

Plausible Modal Rankings:

Guardian Mystic Rogue Seeker Survivor

1 3 1 4 5 2

2 3 1 5 4 2

3 3 2 4 5 1

4 3 2 5 4 1

5 4 1 3 5 2

6 4 1 5 3 2

7 4 2 3 5 1

8 4 2 5 3 1

9 5 1 3 4 2

10 5 1 4 3 2

11 5 2 3 4 1

12 5 2 4 3 1

**Respondent Comments Groups By Cluster**

CLUSTER 1

------------

Guardians have serious bling and they're awesome at what they do, so they're number 1. Seekers also have great cards that guzzle clues and generally provide solid deck building, so they're #2. Rogues have cards that look like a lot of fun (there's bling there too) and they are often good at both clue gathering and fighting, depending on which is needed. Mystic decks feel like they're all the same, so building decks with them is not as much fun. Survivor cards are extremely limited so they're my least favorite.

-\*-

I love the Mystic spells, especially the versatility. Hated Rogues since Skids days, although Jenny is great and Preston is very good fun.

Guardians and Seeker fall very easy into the usable archetypes of Attack and Investigate.

-\*-

I love supporting guardians and seekers. Control focused mistics are also fun.

-\*-

Purple is top just because of Recall the Future and Premonition. Yellow for being weird, Green for extra-actions and Finn. Red for cool, weird interactions at a bargain price. Blue is boring.

-\*-

I don't like playing Rogues, alright? Please don't crucify me! Oh, this is anonymous? Excellent.

-\*-

Simplicity of play and planning.

-\*-

I love spells and magic items

-\*-

Guardian are probably te most rounded IMO. Seekers next, but great at clue gathering.

-\*-

Seeker pool has best card draw & selection; guardian has stick to the plan + stand together + weapons; survivor pool is good but good xp options are less varied (will to survive/true survivor or bust); mystics delve too deep + bag control + David Renfield are good. Rogue pool is harder to build a full game plan around—-its best cards enable great turns (pocket watch, double or nothing, etc) and are valuable to have in the party, but they have a harder time spending actions 2-3 as usefully since some of their best things exhaust (lockpicks).

-\*-

Mystic and Rogue tied for first. Mystic is my prefered and I like how I can stack my deck to be heavy in either investigating and/or combat. Rogue because most get a lot of recources where you can purchase more expensive cards.

-\*-

I feel as though Mystic have the broadest tool kit and be specialise in almost any direction. However my experience is solely limited to two player with my wife and she plays a cloover, so we need someone with bashing power.

-\*-

Matt's response

-\*-

I primarily play a seeker (Daisy)

-\*-

Yellow fits with my playstyle the best

-\*-

I really like all of them, so there's not a ton of distance between them.

-\*-

gameplay style, clear focus on purposes

-\*-

Guardian and Seeker are very straightforward, and I like that. They have a clear objective, and they do it well.

-\*-

While I feel that most classes have merit, the rogue is generally the worst at the core aspects of the game: fighting and clue finding. Evading does not have the punch that killing the enemy foes.

-\*-

I prefer a support / team role, and play for consistency over tricks.

-\*-

Most useful for the group

-\*-

I just looked at options. Mystics have a lot of options in every way, shape or form, and so do Guardians. I just prefer the mystic combos better, since Guardians are pretty bland in that regard. I feel you really can now make different mystic decks, from support to tank and combat master, to main seeking investigator etc. They have everything and even playing one deck a few times is till fun because of so many exp. options. And while their decks are pretty deep, the premise is simple - boost willpower. That leaves them with a nice weakness you have to cover. Guardians have better weapons (more fun) than mystics have combat spells, although Shattered Aeons really gave Mystics a fun new icy option. And maybe I'd like to see a Mystic that wouldn't be pure Mystic if you get me. Some hybrid guy or girl, that's not just using spells and artifacts from the same class over and over again. That's really what they're missing. Guardians are just so great, because they are sooo well balanced imo. It's quite relaxing looking at their options. You have everything from amazing gear, weapons, allies, events that cover literally everything + your friends' asses, awesome skillcards that can also combo, fun and engaging exp. options etc.  But they lack different kinds of investigators. They have options, just some other classes have more. Maybe my least favorite on investigator side. Mystics again are so simple to make in that regard. I gave Seekers 3. because they just have some 0 exp. cards that are just too strong for any class, not just for them. Otherwise I really like Seeker cards theme, maybe even more than Guardian, maybe even my favorite I'd say, but again, Seekers just have so much random stuff and OP stuff (you know what they are). I don't care for balance in a co-op game, OP cards can be really fun, but this stuff really limits their options and sometimes even other classes' options, because not including them just hinders your deck and you know it (example is Shortcut). And that's not good. They have really fun and diverse roster of investigators though. And their experience options are quite game breaking, but in a good way imo. There's seeking, combat, running and evading so much support and combos, really fun and diverse. Rogues have maybe some of my least favorite cards, but they have A LOT of options. They have quite a few very awesome weapons, but they also have SO MUCH cards that are meant for combos and while combo decks are fun, they, in my opinion, are niche, or at least not used in every game. Sometimes you just want a simple deck and Rouges have a limited card pool when you look at it that way (example: no useful combat ally or even asset - there is a new Guardian tarrot card for Jenny and Skids, but they need more imo). They got their quite fresh Lockpicks and the seeker gator and that was an amazing get. But more, Leo breaks their ally pool, because he's just too strong. They also have no pure combat investigators, but otherwise their investigators are really really fun and diverse. They have AMAZING experience options. Maybe the best in the game. And btw, they were my favorite to play before the last few expansions. I love Preston, but again the new cards are very niche. The new seeker agent Joe with 4 combat elevates seekers above Rogues for me in the options in card pool department though. They now have an optional pure combat investigator, while Rogues still don't. Survivors have AWESOME cards, especially investigators are just so fun and weird, but they just lack options in the card pool. You have so many "survive" cards, but they lack anything else strong. Their weapons are quite fun, but there are no heavy hitting options. That for me may be their biggest minus. Lack of experience pure combat options. They have quite a few very strong investigate cards though like Look What I Found and Newspaper 2 exp. And their allies, while strong, are still nicely balanced and quite diverse. They have a million evade options, maybe even too much. It would sometimes be nice to get something else rather than just another evade. These new Track Shoes are pretty cool though. Their skill cards are pretty awesome imo. But still, I feel like they have so much niche cards that only allow some very specific combos, like Rogues, and lack anything else meaningful. They are extremely fun to play though, with all their Survivor specializations like Seeker Urchin, combat Gravekeeper, being dead crazy guy, new athlete runner and evader etc. They may even be my favorite class, but they still lack options in a big way. And they even lack one investigator only available for 15 bucks along a cheaply written book.

CLUSTER 2

------------

survivors da best

-\*-

Guardian just have so many cards that, when looking at them, seem useful. Mystic is my actual favourite class, but it has soo many cards where they went too far with the punishing effects that almost made them useless. Survivor on the other hand has too many events that end up feeling almost the same. Seekers I dont really know, Ive never played them, but everytime I see them looks like they can do many things. And rogue, while it has improved a bit more, I still miss a useful level 1 weapon

-\*-

Difficulty wrapping my head around some classes

-\*-

Mystics are incredibly dependent on their cards.

-\*-

Seekers usually win the game, because the snitch is 150 points

-\*-

Always cards in these classes that I have a hard time cutting. Which means they have the deepest pools marking them the most fun to me

-\*-

I love deck manipulation for seekers, and the flexibility of survivors. I just can't get my head wrapped around mystics.

-\*-

Guardians have a lot of great tools for not just fighting but getting clues. Seeker has the best support so splashing it is great. Rogue and survivor are ties for good splash but survivors card pool is mediocre to me. Mystic aren't bad but I haven't seen it great with others very well. Mystics are good as themselves but really expensive and not great for splash IMO.

-\*-

Survivor have many nice tricks to survive and gather clues.

Guardians because they have the best weapons (flamethrower) and protective tools.

seeker for their upgradable cards and higher ed.

mystic for canceling cards but dont like their only good stat is willpower... rogues seems interesting but never played one.

-\*-

Seekers have action economy (shortcut, pathfinder), card economy, resource economy (Dr Milan + cheap cards) and they advance the game quickly (i.e. discover clues).

Specialist decks are better than generalist decks (in multiplayer, which I play) as they accomplish their goals more consistently, and this favours seekers and guardians.

Stick To The Plan with Ever Vigilant is the most powerful deck element I am aware of.

-\*-

I tend to play builds focused around consistency of succeeding at tests and action efficiency and my rankings reflect the build consistencies in order except rogue who are consistent but just not interesting.

-\*-

Love survivors

-\*-

Seeker is m'y main class

-\*-

Firstly let me preface this with I only own 2 cores and the Dunwich cycle and have yet to play through Dunwich.

Survivor offers the most versatility and always seems to be one of the key factors when beating the odds in most cases as well as enhancing evasion and action economy (survival instinct etc).

Seeker cards are my second favourite due to the amount of utility included within them (i.e. Shortcut, Barricade, Medical Texts, Old Book of Lore etc) as well as allowing you what you need to catapult out in front of the agenda deck with cluevering abilities.

Guardian and Mystic operate on a similar field marginally behind Seeker to me though mystic finds itself slightly higher because of the unique interactions with the encounter deck and rule bending. though in my limited experience they both seem to be the more combat based of the card pools so operate in that same niche for me.

Rogue is unfortunately last but honestly that's just because I haven't had many interactions with them, most of their effects seem too situational to be able to use consistently.

-\*-

I don't like taking the obvious solutions to a problem. I.E: Gun to the face, or Spells for everything.

-\*-

Efficiency at what is needed to complete scenarios - mostly clue getting and combat.

-\*-

Rogue and survivor seem to have the most cards that support each other to suggest a new way of playing. Recursion survivor is fun and different from no money survivor (though you can do both). Rogue has succeed by 2 and rich as options. Seeker has less of that but has the power of untranslated etc cards. Guardians are okay but kind of blah. I can’t see any fun decks to do with mystic. Like, messing with the bag is a cool thing to do in any deck, it isn’t a deck. Playing with doom is a couple cards that need each other but it isn’t a plan for how the game will play out.

-\*-

Definitely hard to rank them, but ranked in order of which I'd most like to have as an off-class

-\*-

I like the consistency in the Survivor card pool and how much individual support there is for the variety of Survivor investigators. Although I like the power level of Mystic cards, it always sucks to have your Rite of Seeking or Shriveling 15 cards down after a full mulligan for them.

-\*-

More scenarios need cloovers and fighters, so all classes outside of Seeker and Guardian are more tricksy and less focused on the goal. This is a hard-enough game as it is!

-\*-

Seeker cards are way too powerful. Rogues are the most fun to play. Survivor cards are super efficient at what they do. Guardian pool is decent but overpriced. Mystics have a few amazing cards, but the rest is pretty meh.

CLUSTER 3

------------

Vaguely from ‘most interactive’ to ‘most straightforward’ with a special mention for the Survivor card pool which has been largely static since roughly core with a few major exceptions.

-\*-

Rogue cards are the most fun for me. More money, more actions, more fun.

-\*-

I seem to like the classes that are less straight-forward than Guardian and Seeker tend to be. (In the sense that they are the archetypical fighters and cluevers.)

-\*-

I like cards that cheat the system and don't depend on leveraging board state

-\*-

Green and purple cards have fun and flashy effects. Blue and yellow cards have more standard effects and narrower deck building options.

-\*-

I didn't play mystics a lot yet

-\*-

The numbers are different depending whether we’re talking theory or practice. In theory the Mystic cards are my favorite, both for flavor and interesting mechanics. In practice I struggle with them and they’re usually the first cut.

-\*-

Combos!

-\*-

I like moneeeey

-\*-

seekers have literally everything, and their cards usually aren't too expensive. rogues have adaptable, streetwise, and really good allies, but they're a bit low in damage output. guardians have really good cards but are limited by how expensive they are. mystic events are amazing, but they are 4th place because level 0 spells kinda suck and are expensive as hell. mystic cards are much better with exp. survivor cards are almost decent. it really sucks that many of their leveled up cards are exile cards, but survivors don't get any extra exp. but in general i find their cards to be lacking in clue-gathering capability and damage dealing. they can turn failures into successes, but that's about it.

-\*-

Guardian is solid and predictable, Rogue is fun. Mystic is challenging, Seeker and Survivor are necessary.

-\*-

THE JANK

-\*-

I really dislike survivors as I simply dont understand how to properly build them (appart maybe Wendy). Even if I have rated mystics 4, I enjoy playing Mystic nearly as much as seeker (which I rated 1) rather than Survivor.

-\*-

I think the rogue theme is portayed very well in their card pool

-\*-

corelation between mechanisms and theme

-\*-

I like big, flashy, ridiculous turns and risky plays, so rogue and mystic are the most fun for me. Guardian and seeker are fine and all, just a bit dry. I don’t understand survivor at all, but I’m happy other people have a thing they like.

-\*-

Rogue and survivor give you interesting and powerful but situational tools that you have to figure out how to apply to the scenario. Mystic and guardian are more about powerful assets that you either draw early and use a bunch or wish you’d drawn earlier but can’t afford now and just commit for icons. Seeker pool makes me sleepy every time I look at it; the only mechanic there I really enjoy is the tome synergies and that’s only with Daisy (Rex, of course, is only played one way).

-\*-

Role-play Value

-\*-

I went for those that get me excited to play or provide thrills or cool combinations as I play (rather than, say, the power of the cards)

CLUSTER 4

------------

Lol moments. We’d all be survivor if we were in this game!

-\*-

The top two were tricky to place; Rogues have fantastically fun combo plays available to them, while I love the 'feel' of many Survivor cards, fighting against fate as hard as they damn well can.

Overall, I find the Survivor pool \*just\* wins out, especially with the excellent Will To Survive and semi-immortal Pete Sylvestre.

Guardians and Seekers are two sides of the same coin; I'd say Guardians edge out, because while a Guardian has a few tools (including the infamous Flashlight) to find clues, Seekers have very few options to take care of enemies.

As with Survivors and Rogues, though, this is close.

Mystics... weeeeell. .. I acknowledge they are arguably the best class, once set up, and while their charges last on their spells. The ability to do everything while testing just one stat can make them very efficient. But... this is the class I enjoy the least, in part due to their over-reliance on their spells. Their solutions never feel more than stopgaps for me, so I find Mystics a hard class to play.

(That won't stop me taking Mystics for a spin though, especially for goodies like Delve Too Deep  )

-\*-

Ability to bounce off Investigators with good resource and action economy, other card pools (including Neutral), as well as capability to start off with no experience — all the way to full campaign with as much power-card investment as possible. Seeker may have 2 of the best cards in the game (Higher Education and Dr. Milan Christopher), but the Seeker card pool as a whole does not stand up. It is both narrow and shallow. Mystic is the most detailed and the most broad, but suffers from undue delay leading to deterioration. Guardian definitely needs to be more broad as well. Both Rogue and Survivor blend well, and provide the necessary breadth to take on challenges while melding with the high-economy Investigators. Rogue has a few 3, 4, and 5 xp cards that push it to the top spot. Even for Lola these statements hold up.

-\*-

On a scale of most interesting vs. most boring. Options for rogues and survivors feel fresh and like there are multiple deck archetypes that are valid. Less so for the seeker and mystic card pools, where I feel like there are more "must include" cards which makes deck building less exciting and more rote.

-\*-

survivor da bass

-\*-

The card pool allows rogue/survivor decks to make specialists. Seekers are all just different flavours of clueverer

-\*-

Personally, I like healing and support best, which guardian does quite well. Survivor has my second favorite card pool, though, for tricks and card recursion.

-\*-

Not much between them but I like guns & ammo, survivor class is cool because it is normies vs horror

-\*-

I really like the guardian cards as i enjoy fighting the monsters that appear in scenarios. Unfortunately my least favorite is mystic. Although they have powerful cards, they often take time to set up and I think that the draw backs on some of their cards are too harsh for what they do.

-\*-

Just what I gravitate towards

-\*-

I like killing monsters

-\*-

Mystics have so much of everything with cool effects added on. Guardian cards are efficient at what they do, but really boring.

-\*-

Survivors feel more unique, guardians kill stuff, seekers feel like you can't win without them (though you really can). Rogues and mystics by default. I like rogues better because of Finn and Sefina being really fun to play.

-\*-

Almost always let my partner(s) play the seekers as I find the rogue and survivor cardpools allow you to fly by the seat of your pants, which I find even more exciting than just being the clue gatherer. Mystic card pool can sometimes take too long to develop. Also many marquis mystic cards flirt around with the doom mechanic which always bites me in the arse. Thirdly, mystic pool doesn't have a strong ally base. What's funny about that is I always play spellcasters in D n D.

Guardian pool is pretty straightforward, one I look at as more of a necessity within the context of the game,but doesn't tug at my heartstrings .

Apologize for the sloppy phrasing in my opine due to a long day. Rankings based on personal preferences only. No meta analysis

-\*-

Agnes

-\*-

Just prefer the in your face monster destruction that Guardian is themed with. Really enjoy that class.

-\*-

Flexibility

-\*-

I love killing things and then getting clues!

-\*-

I like all of them but play seekers least, I also like that guardians can just take the mythos phase to the face

-\*-

I like to be the tank, and with some of the new additions guardians have gotten with getting clues they just shine even more. Mystic I never really play but has so many cards I want if I am playing a dunwich survivor or anyone who can get them, same goes for survivor, very few cards from rogue or seeker makes it into my decks unless I am playing chars like Wendy or Leo who kinda needs them to make them work

-\*-

Number of fun upgrade options in green, base card pool for red, upgrade options in blue, useful upgrades in seeker, purple sucks to play. 

-\*-

I like support / killing role, Carolyn FTW

CLUSTER 5

------------

Weapons are fun.

-\*-

Leo is da alpha male.

-\*-

Red's best

-\*-

There’s more variety of cards that let me build decks I enjoy. As you go down the ranking, there’s less variety or fewer viable sub themes to build around.

-\*-

Seeker is powerful but boring, while mystic getting to punch pack at the game is great, with good support to boot.

-\*-

I enjoy the cardplay in survivor, and the mystic side of things. Seeker cards are generally very powerful. I don’t enjoy playing rogue but there is some good cardplay. Guardian I find less interesting overall as a class

-\*-

Wendy is literally the best investigator in the game

-\*-

I enjoy support cards and interesting, unique effects.

-\*-

I tend to go for lazy game play, and usually guardians allow to smash enemies without worrying too much about strategy. Seekers I love them thematically. Mystics, I never understood how to play them efficiently