Universität Leipzig

Faculty of Physics and Earth Sciences

Max Planck Institute for Evolutionary Anthropology

Department of Human Behaviour, Ecology and Culture

BACHELOR THESIS

Optimizing the linkage and de-duplication of records containing family names using artifical neural networks

Author: Enzo Lima

Matriculation number: 3757382

Reviewers: Dr. Subhajit Paul

Dr. Daniel Redhead

Submitted: 26.08.2021

Acknowledgement

My sincere thanks to Dr. Daniel Redhead for mentoring me throughout the course of writing this thesis. His advice and moments of discussion have provided me with a deeper understanding of research pipelines and academic writing. I am grateful for his lasting patience while teaching me how to write in the R programming language. I also thank Dr. Subhajit Paul for his guidance and feedback, and for giving me inspiration to apply concepts from statistical physics to a social data setting.

I thank Dr. Cody T. Ross for providing me with the rich data set that he has gathered in his own research, and for teaching me to write models in the STAN language. My thanks also go out to Cameron Perot and Nikkin Devaraju, two of my good friends who have always freely shared their knowledge of neural networks with me, and who have given me valuable feedback on early drafts of this thesis.

I thank Dr. Richard McElreath for providing the working space and a supportive environment to work on this thesis. Also many thanks go to the researchers in the Human Ecology department of the Max Planck Institute for Evolutionary Anthropology, for attending my talks during the development phase and giving me helpful feedback about my work. Finally, my gratitude goes out to Sophie Kaube and Jessica Fiegert for helping me to sort out the administration involved in my work at the Max Planck Institute, as well as Isabell Schulthoff for helping me with administration at the University of Leipzig.

Contents

List of Figures

List of Tables

1 Introduction

When data is collected and processed in a non-automated way, errors and aberrations are introduced that make it difficult to extract the true latent information that is represented in the data. This is especially true when considering the types of data collected in the social sciences. Additional complexity arises when data is collected about human subjects, as the only 'uniquely' identifying information about each individual is often their names. These names are typically strings of text with no uniquely defined spelling or meaning across human cultures. Furthermore, given time constraints during data collection, idiosyncratic ways of recording such names are often applied. For example, names may be abbreviated in a way that can be determined by the author, but are ambiguous to any other member of a research project. These problems reduce the potential usability and reproducibility of such data, and may be a cause of errors in downstream analysis regimes.

To address this problem, we develop automated methods for identifying unique cases (such as humans) in noisy social scientific data [i.e., record linkage and deduplication; 2]. In some cases—such as in western, industrialized countries—other, more robust identifying information may be available. This, for example, could be a social security number, mobile telephone number, or passport ID. However, in societies that are geographically isolated, lack technological infrastructure, and do not have strong formal institutions, such robust forms of identifying information may not be available. In these settings, social scientists must rely on the reported names of individuals and their intuition when cleaning their data. We focus on this case, and in doing so, we build upon recently developed machine learning approaches for record linkage and de-duplication [1, 3].

More specifically, we develop a hybrid string matching function that is tailored to strings containing names. Building upon this, we train a neural network that combines the information from several string similarity metrics in order to make predictions about string similarity. To train our neural network models, we conduct a simulation experiment, where we produce a realistic synthetic population of individuals with names as the only clear unique identifier. This synthetic data contains realistic family structures and demographics based on real statistics from electoral census data in Spain (statista[1], INE[2]) and the United States (U.S. Census Bureau[3], U.S. Social Security Administration[4]). Family structures are quantified by probability distributions that predict age differences between partners, number of children sired, and parents' age at birth of their children. To corrupt the simulated names, we develop a further algorithm that changes the spelling of the names to reflect typical errors and abbreviations often observed in social scientific data.

A second simulation experiment is run to compare the performance of our models with models typically implemented for data cleaning in the social sciences. Both of our models outperform standard methods in accuracy, precision, and recall on our simulated data set as well as an empirical data set with noisy records from a population in rural Colombia. Our models enable scientists that are working with noisy name records to perform automated or semi-automated record linkage and deduplication with a high degree of accuracy and reproducibility.

2 Motivation

Across the physical and social sciences, concerns are emerging about the reproducibility of published research results. Here, we define reproducibility as the ability of a different researcher to use the same methods and data outlined in a publication, and achieve equivalent results to those reported in the published work [4]. Despite the importance of reproducibility for the credibility of scientific claims, evidence suggests that the level of reproducibility is lower than desirable across many disciplines [5]. For example, a recent report found that almost 70% of physicists have tried and failed to reproduce the results of a published experiment [6]. Alongside this, over 50% of scientists across the social and physical sciences have at some point tried and failed to reproduce their own published results [6].

[1] https://www.statista.com/statistics/450145/average-age-at-maternity-in-spain-by-autonomous-communitiy/ (accessed on 5 April 2021)
[2] http://www.ine.es/jaxiT3/Tabla.htm?t=1580 (accessed on 5 April 2021)
[3] https://www.census.gov/topics/population/genealogy/data/2010_surnames.html (accessed on 5 April 2021)
[4] https://www.ssa.gov/oact/babynames/limits.html (accessed on 5 April 2021)

Investigation into the reproducibility of published research has largely focused on the analytical reproducibility of results. That is, whether the results of the statistical procedures can be re-run and appear to be equivalent to those reported in a published manuscript [4]. While being able to check and verify any code associated with statistical procedures is important, it is not the only cause of non-reproducible results [6]. The steps prior to analysis, and the broader research pipeline, are rarely shared or reported. Most research that is considered reproducible has only been classified as such based on the availability of code for a statistical model, and a cleaned analysis table.

The provenance of the analysis data is, however, typically unknown. In many cases, this is a product of the data pipeline relying heavily on manual data collection and manipulation—such as the manual assignment of unique identifiers (ID's), and data transformations in a text editor, such as Excel. This has been highlighted by a survey among social scientists in the UK, which found that students and junior researchers identify 'quantitative data collection and analysis' or 'quantitative methods' as subjects they need more training in [7]. Unfamiliarity with automated data collection methods may be a reason why manual data entry happens in the social and biological sciences. Any errors introduced at the earlier stages of data management will impact any published results, yet in the majority of cases, these errors will remain unnoticed given the lack of data and code available [5].

Given the importance of sound and transparent data management practices, there have been recent calls for more reproducible research workflows that extend beyond analytical reproducibility. For example, recent meta-science collaborations have established the FAIR principles: data should be Findable, Accessible, Interoperable, and Reusable [8]. In practice this means that data should have globally unique and persistent identifiers (like the DOI[6] system), that data can be accessed by standardized communication protocols (like HTTP or FTP). Alongside this, data should be able to be read by machines in standardized formats (like CSV, JSON), and contain easy to understand descriptors, and that the data has a clear provenance. These principles have been established due to an observed lack of compatibility or availability of data and associated metadata in published work [8]. The FAIR principles also have an application in machine learning research, with there being recent calls to standardize the descriptions of hyperparameters, used libraries, number of training iterations, type of validation, and evaluation parameters [9].

Open-source software has been developed that aids social scientists with typical data processing and management tasks. Examples of such software are the R packages 'tidyverse'[10] and 'stringdist'[11]. More specifically, the 'stringdist' package contains several of the most popular string distance metrics that allow for easy and fast analysis across large quantities of data. The 'tidyverse' package contains many tools that aid in the transformation and processing of large datasets.

While these guidelines, principles, and tools are a step in the right direction for the reproducibility of published results, there remains a lack of advanced open-source software that helps to facilitate reproducible data pipelines. The available functions are almost exclusively a re-packaging of traditional methods for data management, and do not represent recent advances in more intelligent string matching. Here, we develop an intelligent name matching process that is more accurate than existing methods and allows for transparent provenance of processed data. By further wrapping the algorithm in an easy-to-use function for the R language we endeavour to facilitate compliance with the FAIR principles.

2.1 Problem Description

Typical workflows across the social sciences first collect data on human subjects, and then assign anonymous identifiers to all unique cases (i.e., humans) within the dataset. The problem of assigning unique identifiers within social scientific data is non-trivial as reports of names are generally noisy. For example, a single, unique case whose name is "Christina Roberta Applegate" may be reported in a given survey as "C. Applegate", "Cris", or "Christina A.". Determining which reported name maps onto "Christina Roberta Applegate" becomes more complicated when another unique case shares a similar (or the same) name, and there is no other identifying information. This is espe-

[5] In social scientific research working with human subjects, identifying information about research subjects can not be published on ethical grounds.

[6] A Digital Object Identifier (DOI) is a persistent identifier code, and is one of the international standards described by the ISO. (https://www.doi.org/, accessed on 05 May 2021)

cially salient in relational datasets—such as longitudinal data [12], or social network data [13]—where there are multiple, noisy reports of the names that need to be assigned to the same unique cases. For effective automation of the process of determining and matching these unique cases, software must be robust to such reporting patterns, and other typical errors that appear in reports of names.

2.2 Previous Research

2.2.1 Edit-distance like functions

The term edit distance refers to the amount of operations that are needed to change one string of text into another. The types of allowed operations depend on the specific edit distance function that is used. A commonly used simple edit-distance function is the Levenshtein distance [14]. This metric calculates the number of character insertions, deletions, or substitutions needed to transform one string into another. Due to its simplicity, this metric is extremely fast to calculate, and is effective in finding minor spelling mistakes [15]. A downside of this method is that it can not recognize obvious differences between spelling mistakes, e.g. the substitution "c" to "k" gets the same edit distance as "e" to "m", while the latter is less likely to be a spelling mistake than the former and should therefore represent a greater edit distance. Among edit distance functions the Levenshtein distance is one of the most commonly used metrics in social sciences, but compared to other types of string distance metrics it performs almost universally worse when applied to a variety of name records [15]. Variations of the Levenshtein distance that allow different operations are the Damerau-Levenshtein distance [16], the Hamming distance [17], and the Longest Common Substring [11]. The Levenstein distance can be defined in recursive form as

$$\mathrm{lev}(a,b) = \begin{cases} |a| & \text{if } |b|=0, \\ |b| & \text{if } |a|=0, \\ \mathrm{lev}(\mathrm{tail}(a), \mathrm{tail}(b)) & \text{if } a[0]=b[0] \\ 1 + \min \begin{cases} \mathrm{lev}(\mathrm{tail}(a), b) \\ \mathrm{lev}(a, \mathrm{tail}(b)) \\ \mathrm{lev}(\mathrm{tail}(a), \mathrm{tail}(b)) \end{cases} & \text{otherwise.} \end{cases}$$

Where a and b are the input strings, tail() denotes the string without its first character, and $a[0]$ and $b[0]$ represent the first character of string a and b respectively.

The Jaro-Winkler distance is a variation of the Jaro metric, which is not strictly an edit distance measure because it fails to obey the triangle inequality [15]. In practice, it can be considered an edit distance where only transposition of characters is allowed. The Jaro-Winkler variation has been optimized for use with names by assuming that the first p characters of a name have a lower chance of being incorrectly spelled than the later characters in a string that represents a name [18]. Good results have been obtained on short strings like personal names with this method, compared to other edit distance metrics [15].

2.2.2 Token-based distance functions

Tokens are substrings that are created by extracting parts of a string. Taking n-grams is a well-known method, where all adjacent pairs of characters are extracted as individual tokens (figure 1). The Jaccard similarity is the most basic token-based distance function. It takes the n-grams of two strings as the sets A and B and then returns $\frac{|A \cap B|}{|A \cup B|}$. A commonly used variation of the Jaccard similarity is cosine similarity, which calculates the cosine distance between two n-grams represented as vectors such that

$$sim(A,B) = \frac{A \cdot B}{|A| \cdot |B|}$$

Token-based distance functions have the advantage of not being bound by ordering. The strings "Max Planck" and "Planck Max" will be considered highly similar by a token-based distance function, but completely unrelated by edit-distance like functions.

2.2.3 Phonetic encoding functions

Phonetic comparison functions attempt to encode words in a way that resembles human speech. This is language-dependent and is limited by how phonetically consistent a language is. The best known phonetic encoding algorithm is Soundex [19]. It works by keeping the first letter of the string, deleting all vowels, and converting all consonants into the numbers 1-9. The Soundex algorithm is more context-aware than edit distance and token distance functions, but its use is limited to English and English-like languages [19]. Variations of Soundex are Phonex, Phonix, and NYSIIS (New York State Information Intelligence System) [19]. Double Metaphone is an unrelated phonetic

encoding algorithm that has been designed with better support for non-English words [20].

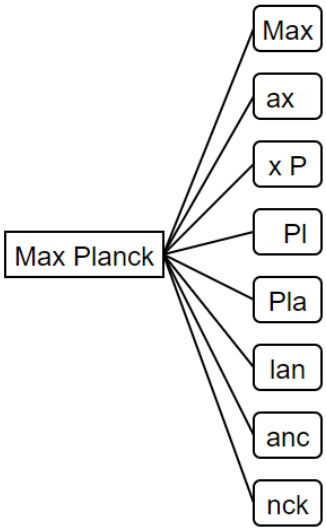


Figure 1: The 3-gram tokens of the string "Max Planck".

2.2.4 Hybrid string matching functions

Hybrid functions combine different categories of string matching functions together in an attempt to combine their strengths. One example is a 'level two distance function', developed by Cohen et al. [15], which builds upon work of Monge and Elkan [21]. In this function two strings are first tokenized and then the maximum similarity of each token between sets is summed, such that

$$sim(A, B) = \frac{1}{\mu} \sum_{i=1}^{\mu} \max_{j=1}^{\nu} sim'(A_i, B_j)$$

where A and B are two strings which are tokenized such that $A = A_1...A_\mu$ and $B = B_1...B_\nu$, and $sim'()$ is some secondary distance function. While there are many different hybrid string matching functions, they generally have in common that they rely on a combination of other string matching functions for their operation and are therefore more computationally demanding than other string matching functions individually.

An example of a more advanced hybrid string matching function is the Heterogeneous Field Matching method (HFM) [1], which attempts to identify strings by tokenizing them into words and then calculating the most likely 'natural' transformations between those words. The transformations that are taken into account are **Equal**, **Synonym**, **Misspelling**, **Abbreviation**, **Prefix**, **Acronym**, **Concatenation**, **Suffix**, **Soundex**, and **Missing**, as shown in figure 2.

These transformations are then used as the features for a machine learning model, based on a Support Vector Machine. This type of machine learning model finds the class boundaries by placing boundary lines in feature space that have the maximum distance from the data points of each class [1]. This model is then used to learn the weights for each transformation and assigns probabilities to the matches.

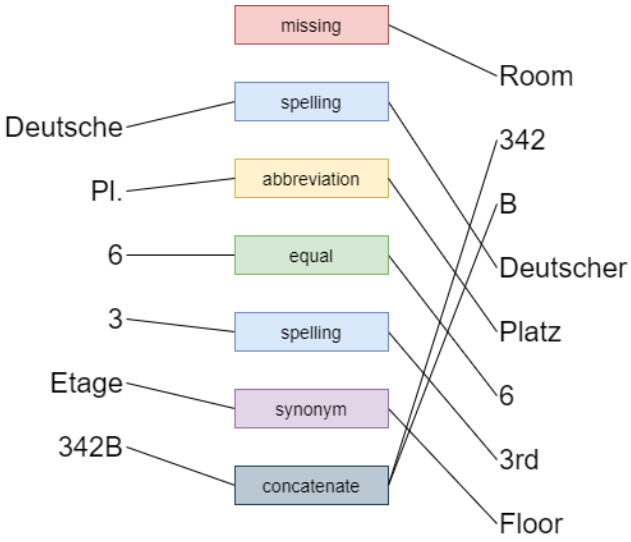


Figure 2: A transformation graph as described by Minton and Nanjo[1].

The discussed string matching functions provide us with numerical information about some property of the two strings that are compared. For edit-distance like functions, it is the number of operations needed to turn one string into the other, for token-based distance functions it is the amount of tokens the two strings have in common. We will now turn to the methods that we can use to harness this information for the purpose of matching names. Furthermore, while some of these functions are usually described as "distance functions", from now on we will refer to each of them as "similarity functions", since we use the normalized output of these functions, scaled so that 0 represents no match and 1 represents two identical strings.

3 Methods

To develop our model, we need a technical framework, training and validation data, and features (numeric values that describe the data's properties). For greater depth and clarity of our technical framework, we have limited the focus of our methods to neural networks.

Neural networks are 'black box' models where an input is fed into the model, which then processes it by applying a set of weighted non-linear transformations to calculate an output [22]. By comparing this predicted output to the expected output (the 'ground truth') and calculating the gradients over each non-linear transformation, the weights of a model can be updated ('trained') to give an output that approximates the expected output as close as possible. The term 'black box' refers to the fact that it is difficult to interpret the weights of this model, and this is almost always not even attempted [23]. For the vast majority of applications of neural network models, practitioners only interpret the relevant inputs and the outputs of the models.

The performance of a neural network improves with the amount of well-performing training data that is available to it [22]. Since examples of name misspellings are not easily available in large quantities, and existing identifying data in social sciences are often restricted due to ethical considerations, we opt to create a simulation that can produce realistic names and family structures. Furthermore, we develop a second simulation that attempts to corrupt these names according to the patterns observed across existing data.

To prepare our training data for the neural network, we need to calculate descriptive features that can translate the properties of each training sample into a set of numeric values. We use existing string similarity functions and develop a new hybrid string similarity function to generate the input data for our neural network.

3.1 Simulation

To generate training data for our model, we create a simulation algorithm that produces a population of individuals that are assigned realistic names[7]. First, we outline the components of our algorithm that are necessary to generate a realistic population structure. We then overview how our algorithm generates names, and introduces corruptions to these names. This simulated data will serve as the ground truth, and allows us to explicitly capture the generative process that produces name reporting errors in social scientific data.

3.1.1 Population Simulation

Our population simulation algorithm initially generates a random set of n individuals, with birth years following a Poisson distribution with a width of one generation. For this application we took the width of one generation to be 31 years, which is based on the average age of first motherhood in Spain[8]).

Algorithm 1 People Table Simulator

Require: USA census data
  Generate initial table of random people
  Generate initial table of random marriages
  Generate initial table of random external people
  for $iteration = 1, 2, \ldots, n$ **do**
    Generate random alphanumerical id
    Select parents from marriage table
    if childname = "paternal" **then**
      lastname = father lastname
    else if childname = "maternal" **then**
      lastname = mother lastname
    else if childname = "paternalmaternal" **then**
      lastname = father + mother lastname
    else if childname = "maternalpaternal" **then**
      lastname = mother + father lastname
    end if
    if ethnicitymother = ethnicityfather **then**
      ethnicity = ethnicitymother
    else
      ethnicity = mix
    end if
    Select gender
    Select firstname
    Select birthyear
    if random(TRUE/FALSE) **then**
      Select single person
      Add new entry to marriage table
    end if
    add new entry to people table
  end for
  $b = b + y_c - max(b)$

[7] The full code for this function can be found in appendix C.
[8] https://www.statista.com/statistics/450145/average-age-at-maternity-in-spain-by-autonomous-communitiy/

After the initial list of n people has been generated, a loop is run for n iterations (algorithm 1), generating one new person per iteration, which produces 'offspring' from the initial list of people (or population register). Several measures are taken to prevent unrealistic family structures. First, we specify an incest taboo that prevents immediate kin from being selected as possible partners. That is, within our algorithm we include a parameter that determines the maximum allowed relationship coefficient Wright [i.e., the 'inbreeding coefficient'; 24]. For every iteration of the algorithm, a relationship coefficient is calculated between the generated person and the people in the singles database (which is a table that contains all couples).

Alongside this, we specify a Poisson distribution to calculate the chance of couples having more children, while taking the couple's current number of children into consideration [25]. The Poisson distribution allows us to describe the probabilities for each successive child a couple may produce. Every new entry is then appended to the list of people, including parentage information. Finally, the original n random people are removed from the table and only the offspring is returned as output.

3.1.2 Name Simulation

We obtained a corpus of 263,157 forenames and 162,253 surnames from the U.S. Census Bureau[9], and the U.S. Social Security Administration[10]. The U.S. Census was chosen because the U.S. population has ancestry from many different countries[11], and therefore contains names from a variety of different languages and ethnicities. This provides us with a source of diverse names to use as training data. During each iteration, an individual entering the population was assigned a gender and ethnicity based on their first and last name. Parentage is left empty and random alphanumerical IDs are assigned to each simulated person.

3.1.3 Simulation of Reporting Errors

Now that we have simulated a realistic population, and know their true names, we need to incorporate the common reporting errors that are observed across social science data. To do this we develop an error simulation algorithm (algorithm 2) that takes a full name and its phonetic (IPA[12]) transliteration as input, and mutates it to a misspelled version according to several sets of probabilities[13]. For the phonetic transliteration we use the Grapheme2Phoneme[14] web service [see 26, for further details].

Algorithm 2 misspelling generator

Input string of full name
Input string of ipa symbols
With P_a, abbreviate one or more of the names
With P_m, delete one or more of the names
With P_p, generate a phonetic transliteration
With P_e, introduce random edit distance errors
Return misspelled name

Our algorithm then takes the probabilities for a token to be abbreviated: $P_a = (P_{a1}, P_{a2}, P_{a3})$, to be missing: $P_m = (P_{m1}, P_{m2}, P_{m3})$, to be phonetically misspelled: $P_p = (P_{p1}, P_{p2}, P_{p3})$, and to contain random errors: $P_e = (P_{e1}, P_{e2}, P_{e3})$. Each probability is a vector where the first element refers to the first word of a name, the second element refers to the middle words of a name, and the last element refers to the last word of a name. For example, outputs for the name "Lester John Sixsmith" could be "Lester J Sicksmit", "L Sissmit", "John S", or "L J Sixsmith". We perform multiple passes to simulate the multiple surveys on which a latent individual might be referenced. Using this noisy simulated data, we generate the features that we need to train and validate our name matching model.

3.2 Features

3.2.1 A Name Similarity Function

To generate informative data for our neural network, we create a new, asymmetric string similarity func-

[9] https://www.census.gov/topics/population/genealogy/data/2010_surnames.html (accessed on 05 April 2021)
[10] https://www.ssa.gov/oact/babynames/limits.html (accessed on 05 April 2021)
[11] https://www.census.gov/data/tables/2000/dec/phc-t-43.html (accessed on 05 April 2021)
[12] International Phonetic Alphabet.
[13] The full code for this function can be found in appendix D, or at https://github.com/errslima/EnzoBachelorThesis.
[14] https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface/Grapheme2Phoneme (accessed on 11 May 2021)
[15] The full code for this function can be found in appendix E, or at https://github.com/errslima/EnzoBachelorThesis.

tion that is tailored to name matching[15]. Our function takes a dyad of an observed name and a reference name as input. Each name is then split into single word tokens. Given a dyad "JN Smit T" and "John Alexander Smith Taylor", the tokens are "JN", "Smit", and "T" for the first name and "John", "Alexander", "Smith", and "Taylor" for the second name. These tokens then form the axes of a comparison matrix, as seen in table 1.

	JOHN	ALEXANDER	SMITH	TAYLOR
JN	71	20	0	0
SMIT	0	0	89	20
T	0	0	0	50

Table 1: A name comparison matrix.
The best matching subdyads have been highlighted. Their sum is the total score given to the dyad of "JN Smit T" and "John Alexander Smith Taylor".

The vertical axis of the matrix contains the tokens from the observed name, and the horizontal axis of the matrix contains the tokens from the reference name. Each cell of this matrix then represents a 'subdyad', a pair of two tokens. Next, our function determines whether any subdyad should be considered (1) an abbreviation, (2) a misspelling, or (3) an unrelated pair of tokens. A score of 50 is assigned to abbreviations, a score of 0 is assigned to unrelated tokens, and a similarity score between 0 and 100 for each misspelling (algorithm 3) is calculated by some string similarity function, which we refer to as the 'core function' of this name similarity function.

The function then calculates the sums of the scores of the best matching subdyads. A combination of subdyads (combination of cells in the matrix) is allowed if each observed token forms a dyad with at most one reference token, and vice versa. Furthermore, all subdyads d_n in row r_n and column c_n must be positioned such that subdyad d_{n+1} is positioned in row $r_{n+1} > r_n$ and column $c_{n+1} > c_n$. This precludes the possibility of swapped tokens, and ensures that one token is not matched up to multiple other tokens. An example of can be seen in table 1. The combination with the highest sum is then normalized by the highest possible score for the observed name and a similarity score between 0 (no similarity at all) and 1 (all tokens in the observed name could be completely matched in the reference name).

Algorithm 3 Name Similarity

```
Input observed name
Input reference name
Separate names into word tokens
Create similarity matrix
Assign scores
if word1 has length 1 and = first letter of word2 then
    score = 50
else if word2 has length 1 and == first letter of word1 then
    score = 50
else if word1 has length 1 and != first letter of word2 then
    score = 0
else if word2 has length 1 and != first letter of word1 then
    score = 0
else
    score = [core function]
end if
Select combination with highest score
Normalize
```

3.2.2 Information and Performance

In order to better understand what each string similarity function tells us about the strings that are being matched, we plot a histogram for the normalized scores that are assigned to matching and non-matching strings. By measuring the overlap between these two histograms, we can measure how well each function can separate matches from non-matches. The graphs for a selected number of string similarity functions that we have tested can be found in figure 3. In the graph we can see how the name similarity function with a Jaro-Winkler core or with a Levenshtein core performs good separation between matches and non-matches, whereas the 3-gram similarity function shows a lot of overlap between matches and non-matches.

3.2.3 Blocking

It is often not computationally practical to calculate all features for all input samples when large amounts of data need to be processed. In order to reduce the processing time, we can use a computationally light metric

that can group all matches by a given threshold value. This technique is usually called "blocking" [15].

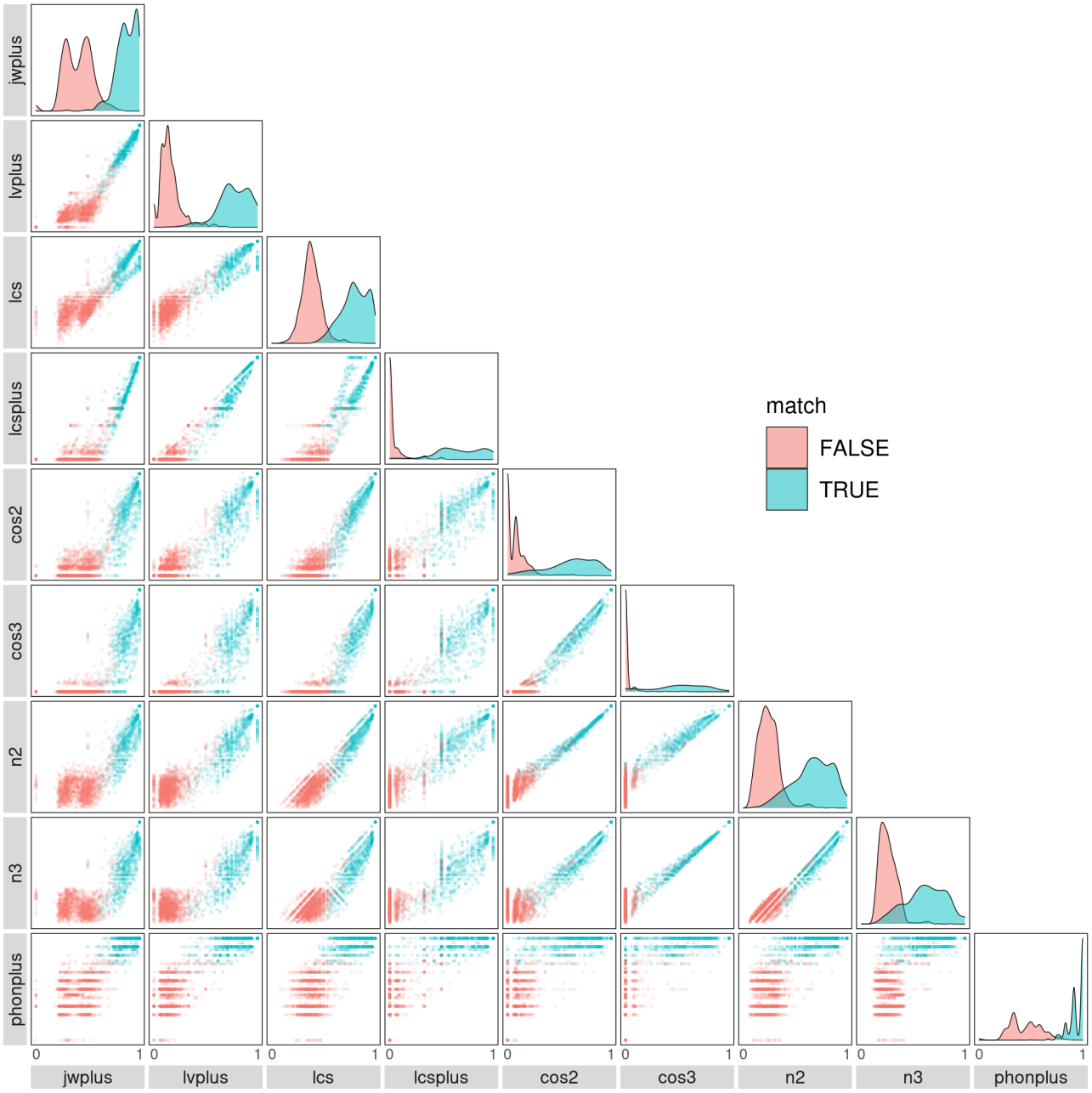


Figure 3: A system of pair plots for selected features.
jwplus = name similarity function with a jaro-winkler core, lvplus = name similarity function with a levenshtein core, lcs = longest common substring, lcsplus = name similarity function with an lcs core, cos2 = 2-gram cosine similarity, cos3 = 3-gram cosine similarity, n2 = 2-gram similarity, n3 = 3-gram similarity, phonplus = name similarity function with a soundex core.
The graphs on the diagonal show where the matches and non-matches overlap in value for each feature. Less overlap represents a more informative feature.

To find the optimal string similarity function for this, we measure the speed with which each function is calculated. The results are displayed in table 2. Furthermore, we determine the blocking performance for each feature by measuring what fraction of non-matching dyads has a lower value for that feature than the lowest value of the matching dyads (i.e., the threshold)[16]. All non-zero results are displayed in table 3. Based on these results, we have chosen to use the Longest Common Substring (LCS) feature as blocking parameter with threshold value 0.35.

Feature	Speed
n-gram cosine	$\pm 10^5$ dyads/s
n-gram jaccard	$\pm 10^5$ dyads/s
Jaro-Winkler	$\pm 5 \times 10^4$ dyads/s
Levenshtein	$\pm 3 \times 10^4$ dyads/s
LCS	$\pm 3 \times 10^4$ dyads/s
n-gram similarity	$\pm 2 \times 10^4$ dyads/s
Soundex	$\pm 8 \times 10^3$ dyads/s
Name Similarity	± 600 dyads/s

Table 2: The processing speed of features, as measured on a single core 2.10 GHz processor.

Feature	Blocking (±5%)
NS (LV)	88% (0.27)
LCS	59% (0.40)
NS (Jaccard-1)	51% (0.22)
Levenshtein	51% (0.17)
NS (Cosine-1)	45% (0.32)
1-gram similarity	36% (0.50)
NS (JW)	26% (0.30)
1-gram cosine	23% (0.39)
1-gram jaccard	19% (0.29)
2-gram similarity	17% (0.15)
Jaro-Winkler	12% (0.40)
3-gram similarity	3% (0.14)

Table 3: The relative amount of non-matching dyads that falls outside of the range of values of matching dyads.

The threshold value is added in brackets after the percentage.

3.2.4 Ground Truth

Two forms of predictive models that we will consider here are classification models and regression models (more specifically, logistic regression). Classification models (classifiers) assign probabilities to different classes, where the most probable class is then chosen as the prediction [27]. Logistic regression models (regressors) attempt to describe a continuous dataset using a multivariate logistic equation [27]. This makes regression models suited for situations where continuously scaled output is desired.

Technically we are dealing with a classification problem–we want to know if a dyad of names represents a match or not. However, we are also interested in the likelihood that a dyad represents a match. This makes our problem more suited for regression analysis.

To emulate the noise observed in social science data, the training data that we simulated comprises many dyads that, while having comparatively similar features, have opposite classifications. If we were to train our model as a classifier, such data points could result in our model learning 'patterns' that don't exist, and essentially attempt to enclose each individual data point with its own classification. In practice, we want to determine the probability that a dyad with certain features is a match based on the fraction of samples in the training data with similar features that represent a match. To do this, we calculate the probability of each training sample being a match by weighing the k nearest neighbours (kNN) of each training sample [27]. Then, instead of training our model as a classifier, we will train it as a regressor and use the weighted probabilities of each training sample as our ground truth.

Given this, we define the probability ρ of a training sample representing a match as

$$\rho_i = \frac{N_i}{\sum_j q_{ij}}$$

$$q_{ij} = 1 - \sqrt{d_{ij}}$$

$$d_{ij} = \sqrt{\sum_k (f_{ik} - f_{jk})^2}$$

where N_i is the number of matches in the region around the i^{th} training sample, q_{ij} is the weighted influence from the the j^{th} nearby sample on the i^{th} training sample, as calculated over all features f_k ($k = 1...N_{features}$), such that a distance of 0 is equal to a weight of 1 and a distance of 1 (the longest possible distance in our feature space) is equal to a weight of

[16] The code we used to determine the blocking performance of the features can be found in Appendix H, and is available online at https://github.com/errslima/EnzoBachelorThesis.

0. We take the square root of the distance term d_{ij} to reduce the 'smoothing' of patterns that this procedure causes. Furthermore, we only compare samples that have a maximum distance of 0.2 to the i^{th} training sample.

3.3 Machine Learning and Neural Networks

A neural network is a machine learning model that uses non-linear transformation models called 'neurons' to perform calculations on the input features [22]. These neurons have weights or biases, which are the parameters that are being trained when the network is learning. Furthermore, they are organized into fully connected 'hidden layers' (see figure 5). It is difficult, if not downright impossible to interpret the weights and bias in such a hidden layer[23]. The only variables that can be realistically interpreted are the input features and the output. For the current research, the output is a number between 0 and 1, where 1 represents complete certainty about a match, and 0 represents complete certainty about a non-match. The neural network 'learns' by feeding the input features into the hidden layers, and then comparing the output to the ground truth. This comparison is some loss function that determines the difference between the two, after which the gradient for each weight and bias can be calculated by differentiation.

For the algorithm let w_i and b_i be the weights and biases on i'th layer. Furthermore, σ is the sigmoid function such that

$$\sigma(\theta) = \frac{1}{1+\epsilon^{-\theta}}$$

and x is the input layer of features, which is an $m*n$ matrix with m observations and n features. $\hat{y}$ represents the prediction made by the neural network. This prediction is compared to the ground truth y using a loss function c. The loss function determines how far the prediction is from the ground truth [22].

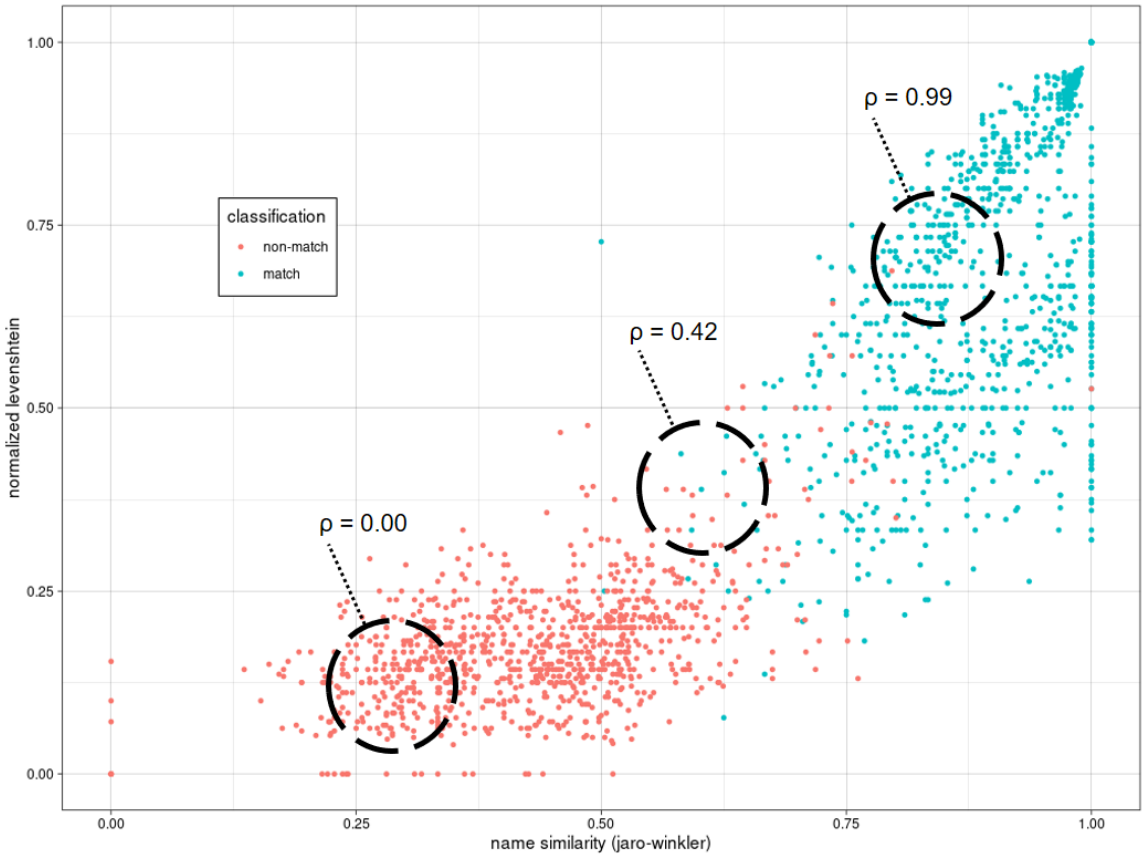


Figure 4: A pairs plot showing the distribution of matches and non-matches over two features. ρ represents the density of matches vs. non-matches at the training sample in the middle of the circle.

As our neural network does regression, we use the mean squared error (MSE), which is defined as

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

where $\hat{y}_j$ denotes the prediction for the j'th input.

If we were to use the network as a classifier instead of a regressor, we could use a cross-entropy loss function. This function is defined as $c = H(\hat{y}) + D_{KL}(\hat{y}||y)$ where $H(\hat{y})$ represents the entropy of the predictions and $D_{KL}(\hat{y}||y)$ represents the Kullback-Leibler divergence between $\hat{y}$ and y[28].

Now that we know the loss, we calculate the gradient for each neuron, so that we can update the weights and biases to minimize the loss. This gradient calculation is done automatically using an *autograd* function in 'Torch for R' [29]. The calculation of the gradient is done for every iteration t, where one iteration is one forward pass to produce a prediction, and then a back-propagation pass to calculate the gradients. Finally, we can set a 'learning rate' to adjust the magnitude of the update done during each iteration. A higher learning rate will cause the network to learn faster, but it may not find the optimal solution. A lower learning rate will allow the network to find better solutions, but setting the rate too low will cause the computation to become intractable. Through experimentation we have found that a learning rate of $r = 0.1$ is the lowest learning rate that allows the model to converge in a reasonable amount of time.

Taken together, our neutral network model can then be defined[17] as

$$l_1 = \sigma(w_1^T x + b_1)$$
$$\hat{y} = \sigma(w_2^T l_1 + b_2)$$
$$c = \frac{1}{n}\sum_{m}^{n}(\hat{y}_m - y_m)^2$$
$$dw_i = \text{autograd}(c)$$
$$w_i^{t+1} = w_i^t - r \times dw_i$$
$$b_i^{t+1} = b_i^t - r \times db_i$$

For a neural network with the amount of features and neurons that we use, convergence usually occurs within 10,000 iterations.

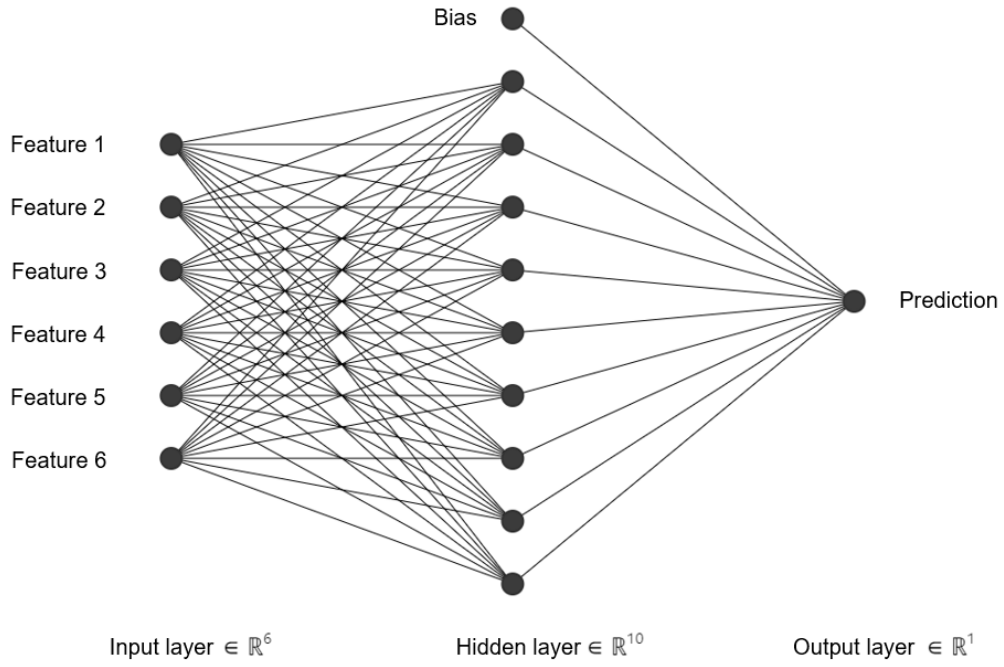


Figure 5: Visual representation of a neural network.
The bias for the hidden layer is the top neuron which has no input.

[17]The full code that was used to train our neural network can be found in appendix F. The code that was used to make predictions using our trained model can be found in appendix G. , Both are available online at https://github.com/errslima/EnzoBachelorThesis.

3.4 Semi-Automated Matching

Fully automated record linkage and de-duplication models have the advantage of scalability to large datasets and are able to run without pause for extended periods of time. However, these automated approaches do not have the same semantic intuition that humans do [8]. Even with our specifically tailored neural network model, a non-trivial amount of false positives or negatives will inevitably be returned.

To solve this problem, we can combine the processing speed of a machine with the semantic intuition of a human by running the models in semi-automated mode. This means that the model returns the top n most likely matches and presents them in an interface to the user. The user can then select which of the matches seems to be the correct one, or specify that none of the options are a good match. While this will slow down the matching process, it makes up for the extra time by resulting in increased accuracy. Any true matches that were not returned in the top n most likely matches will end up not being matched, and may be manually reviewed later.

To calculate how well the models perform in semi-automated mode, we have chosen $n = 4$ and assume that the person performing the matches will always pick the right match if it is presented among the four choices. Furthermore, we assume the person doing the matching will enter "no match" if the right match is not presented among the four choices.

4 Model Validation

4.1 Hyperparameters

To determine the optimal number of neurons and hidden layers in our neural network, we have created neural networks with zero, one, and two hidden layers, with a range from 8 to 36 neurons in each hidden layer. For each of these configurations, 100 distinct neural networks are run for 1000 iterations, after which the error is calculated as the fraction of wrong predictions from the total predictions the network has made. From these 100 networks, the average error is calculated for every configuration.

As shown in figure 7, a single hidden layer produces a much lower error rate than zero hidden layers, which indicates that there is no linear solution to our problem. Adding a second hidden layer does not have any significant benefit. Furthermore, the error rate does not seem to decrease further when we add more than 21 neurons to each layer.

4.2 Simulated Data

To test our model we have simulated 9800 names, with 30 misspellings generated per name. Out of the 294,000 resulting 'observed' names, we have randomly selected 500 observed names along with their correctly spelled names, which will serve as our reference names. There were 259 unique reference names. Out of these names, we selected 209 reference names as our 'internal' population (people whose name we know and exist in the reference list), with the remaining 50 reference names becoming our 'external' population (people who are not in the reference list). By using an external population, we will be able to measure how well our model separates matches from non-matches.

For performance assessment we have used the metrics 'precision' (the number of true positives as a fraction of all predicted positives) and 'recall' (the number of true positives as a fraction of all actual positives)[18]. Our model calculates the matching score (a number between 0 and 1) for every dyad of observation and reference, and then selects the reference name with the highest score. If the score of the selected match falls below a threshold, the match is deleted. If the score falls below that threshold, it is kept. We have then calculated the precision and recall for 100 different thresholds of match scores, ranging from 0 to 0.99 in increments of 0.01. The resulting precision-recall pairs have been plotted in a precision-recall graph.

We compare our model with common string matching functions, and the results are plotted in figure 8. The graphs seem to have different domains, this is an effect of the precision-recall graph actually being a three-dimensional graph that only shows two of its dimensions. The third dimension is the threshold value, with values ranging from 0 to 0.99. This third dimension can be visualized on a z-axis, which shows that the right extreme value of each graph represents 0% threshold and then moves to the left extreme value, which represents a 99% threshold.

[18] For a more detailed explanation of the terms 'precision' and 'recall', refer to Appendix A.

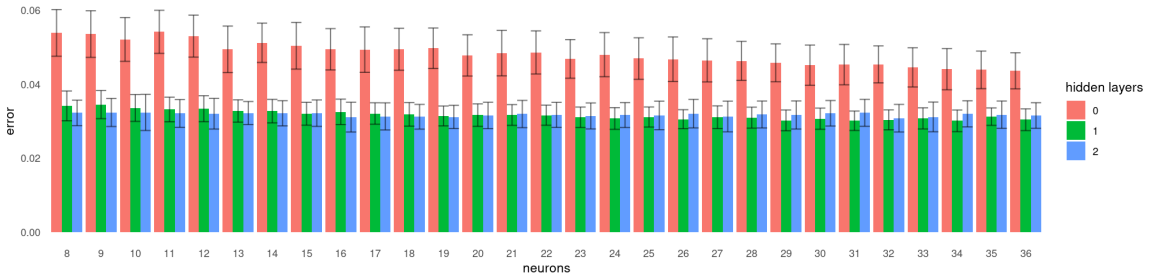


Figure 6: Bar plot showing the mean accuracy of a neural network after 1000 iterations for a varying number of hidden layers and neurons.

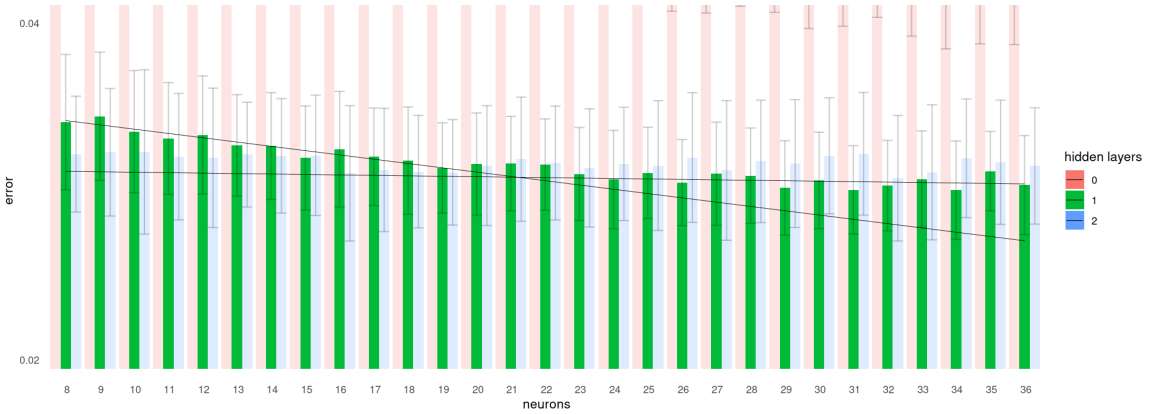
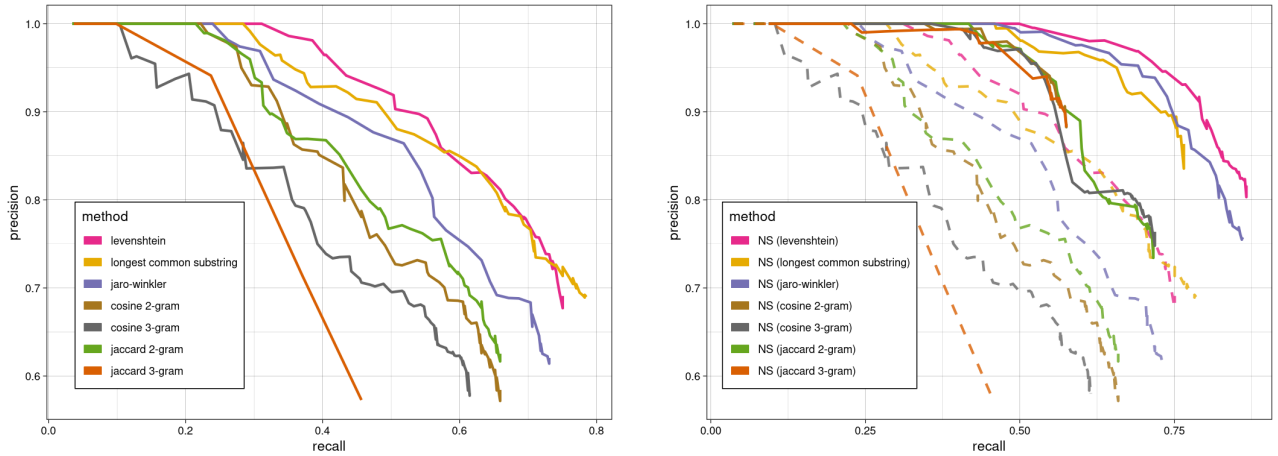


Figure 7: Bar plot showing the mean accuracy of a neural network after 1000 iterations for a varying number of neurons, with one hidden layer.

We choose the optimum amount of neurons as the intersection of a linear fit to the first ten neural networks with a linear fit of the last ten networks.



(a) (b)

Figure 8: Two line charts comparing the performance of common string similarity functions and the new name similarity function.

Performance of the common string distance functions are shown in figure (a) and the name similarity function with different core functions is plotted in figure (b). The dashed lines in the background of graph (b) show the corresponding string similarity functions from graph (a) for comparison.

The name similarity function that we have developed clearly outperforms the existing methods, particularly when using a Jaro-Winkler or Levenshtein core function. At 50% recall the name similarity function is able to achieve almost perfect precision on the simulated data.

Finally, we have trained several neural networks with the metrics from the string matching functions and the name similarity function as input. The performance results have been plotted in figure 9. An overview of the features used in each neural network model can be found in table 4.

	COS2	COS3	LV	LCS	NS (jw)
Model 1	x	x	x	x	x
Model 2	x		x	x	x
Model 3		x	x	x	x
Model 4			x		x

Table 4: An overview of the features that each neural network model uses.

Every model was trained with 22 neurons and one hidden layer.

We can see that the neural network models outperform the name similarity function in the high-recall region, but underperform in the low-recall region. In other words, the neural networks are better at finding true positives, but at the cost of also introducing a substantial amount of false positives.

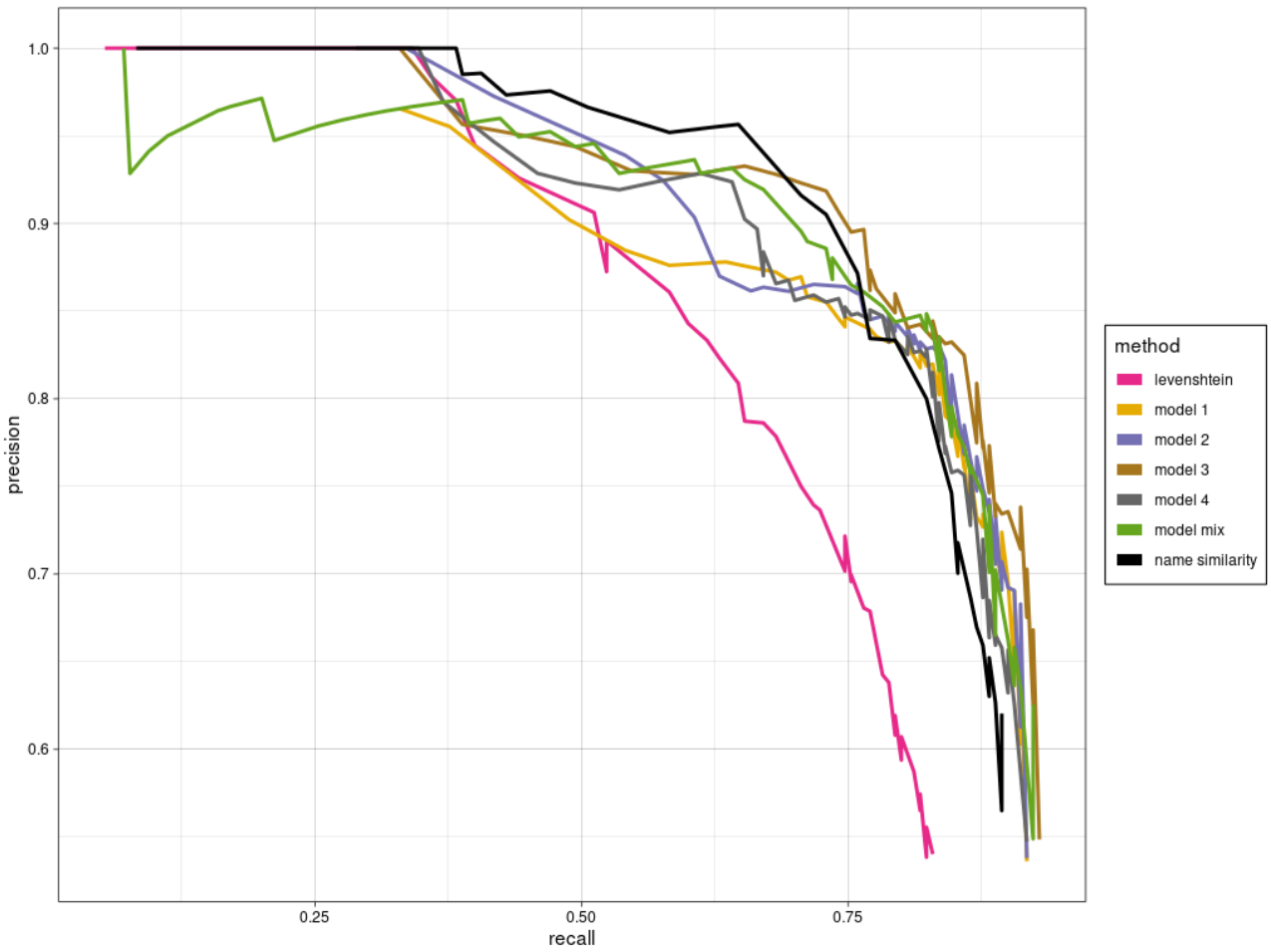


Figure 9: Performance of four different neural network models on simulated data. The name similarity function was run with a jaro-winkler core function.

5 Empirical Application

5.1 Colombia Fieldsite

Our first data set is obtained from survey interviews that were administered in an Afro-Colombian community in rural Colombia[19]. The population of in this data set has an average relatedness of 0.008[20] and unknown name descent (the system by which children obtain their last name) [30].

5.1.1 Data collection

The data relevant to the current research was drawn from recently undertaken census, and social network surveys. Census data were collected by Dr. Cody Ross and several interlocutors, who conducted interviews with a representative from each household—who were usually the adult heads of the household. Social network data were collected by asking these household representative a battery of 'name generator' questions [31, 32], where respondents were, for example, asked "*Who would you go to for a loan of a week's wages?*". The census data was subset to include only name information data, which was used as our reference list of names. The social network data (i.e., the names of the individuals that were nominated during questioning) as a list of candidate names that needed to be matched to the reference list of names. Because the collected data contains confidential information, only the calculated features can be published, and not the original names.

The noise in this data can largely be considered as resulting from the issues that have been detailed in our problem description. However, additional noise was introduced to these data as names recorded in the social network interviews also included descriptive notes written in brackets. An example of this would be "John (brother of Lisa)". To examine model performance with only name information, we filtered our these descriptive notes. For completeness we have also included the results of our model on the non-filtered

5.1.2 Results

A comparison of the name similarity function with common string matching methods shows that it outperforms them in both precision and recall at every threshold level, as can be seen in figure 10. Decent results are achieved by token-based similarity functions, but all of the edit-distance based functions show poor performance. The worst performing metric is the Levenshtein similarity function, which is a commonly used string matching method [14]. Compared to the best performing common string matching metric, the name similarity function shows on average 20% more recall and 25% more precision in its predictions. Compared to Levenshtein matching, these numbers go up to 400% and 300% respectively.

The neural network models that we have trained seem to slightly underperform the stand-alone name similarity function. An interesting observation, displayed in figure 11, is that when the outputs of neural network model 4 are averaged with the outputs of the name similarity function (the graph labeled "model mix"), the resulting predictions seem to outperform the name similarity function. The neural network model uses the name similarity functions as one of its features, so by averaging the two we are essentially increasing the weight of the name similarity function on the final prediction. It is possible that the neural network inadvertently assigns too much predictive weight to the other features.

In figure 13 we can see comparisons between the performance in fully automated mode versus semi-automated mode. To calculate the performance in semi-automated mode, we have selected the top four most likely matches from every prediction. If the correct match was in one of those four, we assume the person doing the matching would select the correct one. For this comparison graph, the recall performance is the most important metric. The precision metric becomes less important because the person who performs the semi-automated matching can deny the match when it is presented. From the graph we can see that the name similarity function is slightly outperformed by the 3-gram Jaccard similarity in recall. Our neural network model performs better than the name similarity function, and performs equally well as the 3-gram Jaccard similarity.

[19] A sample of this data set can be found in appendix I. The full data set can be found online at https://github.com/errslima/EnzoBachelorThesis.

[20] calculated by taking the weighted average of coastal and inland Afro-Colombians

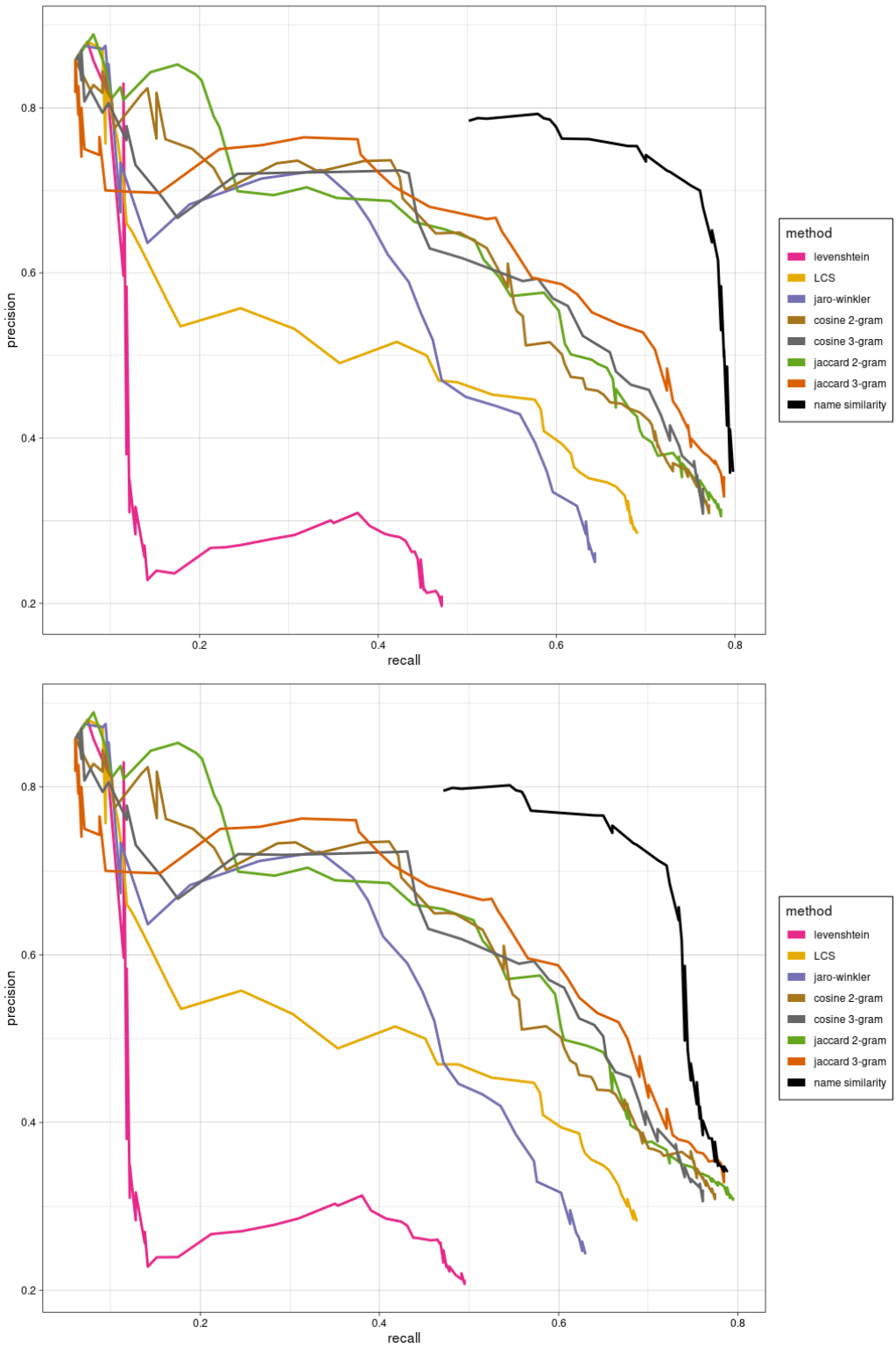


Figure 10: Relative performance of features on prefiltered (top) and unfiltered (bottom) data from the Colombia fieldsite.

The name similarity function was run with a jaro-winkler core function.

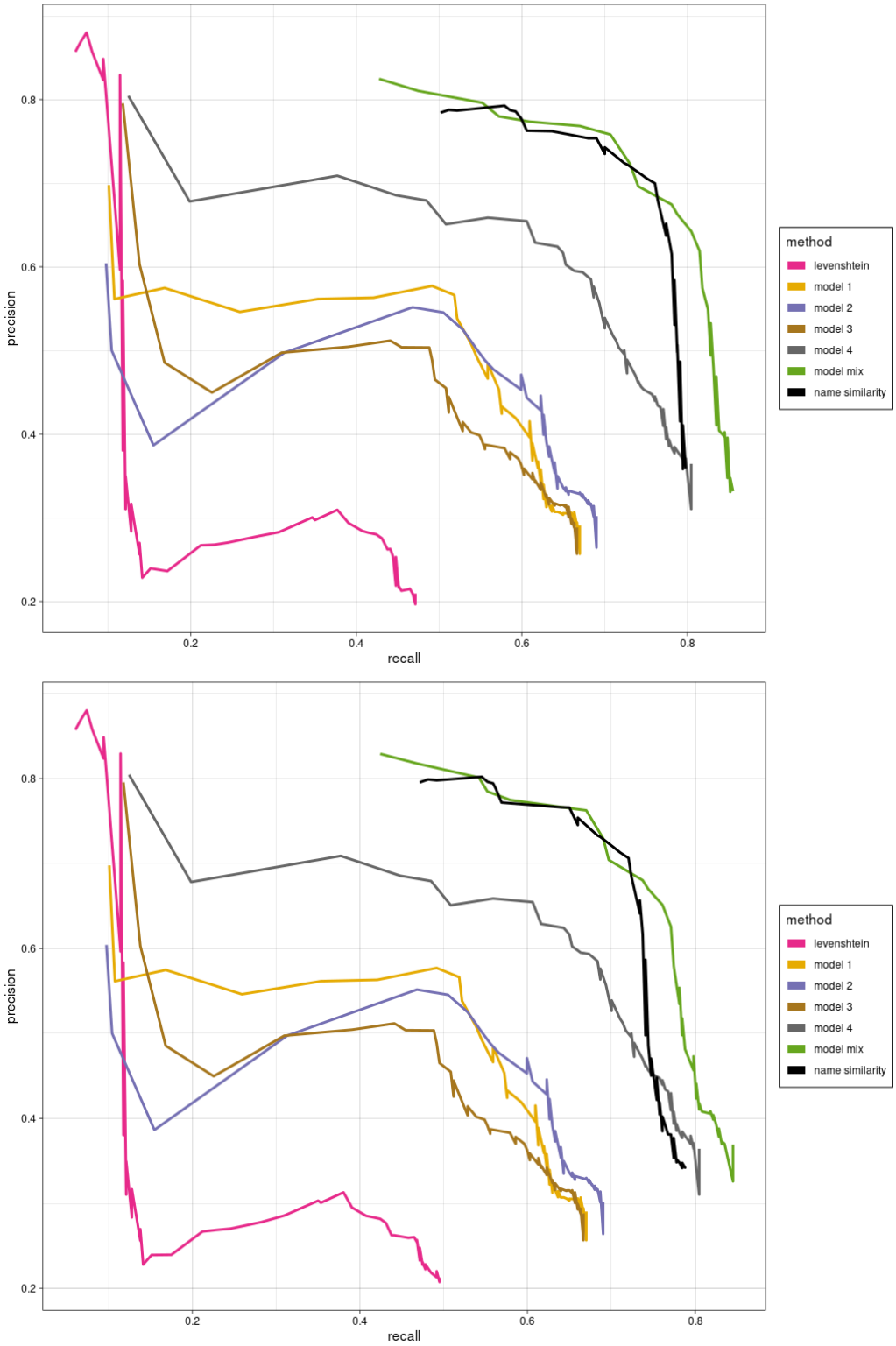


Figure 11: Relative performance of neural network models on prefiltered (top) and unfiltered (bottom) data from the Colombia fieldsite.

The name similarity function was run with a jaro-winkler core function.

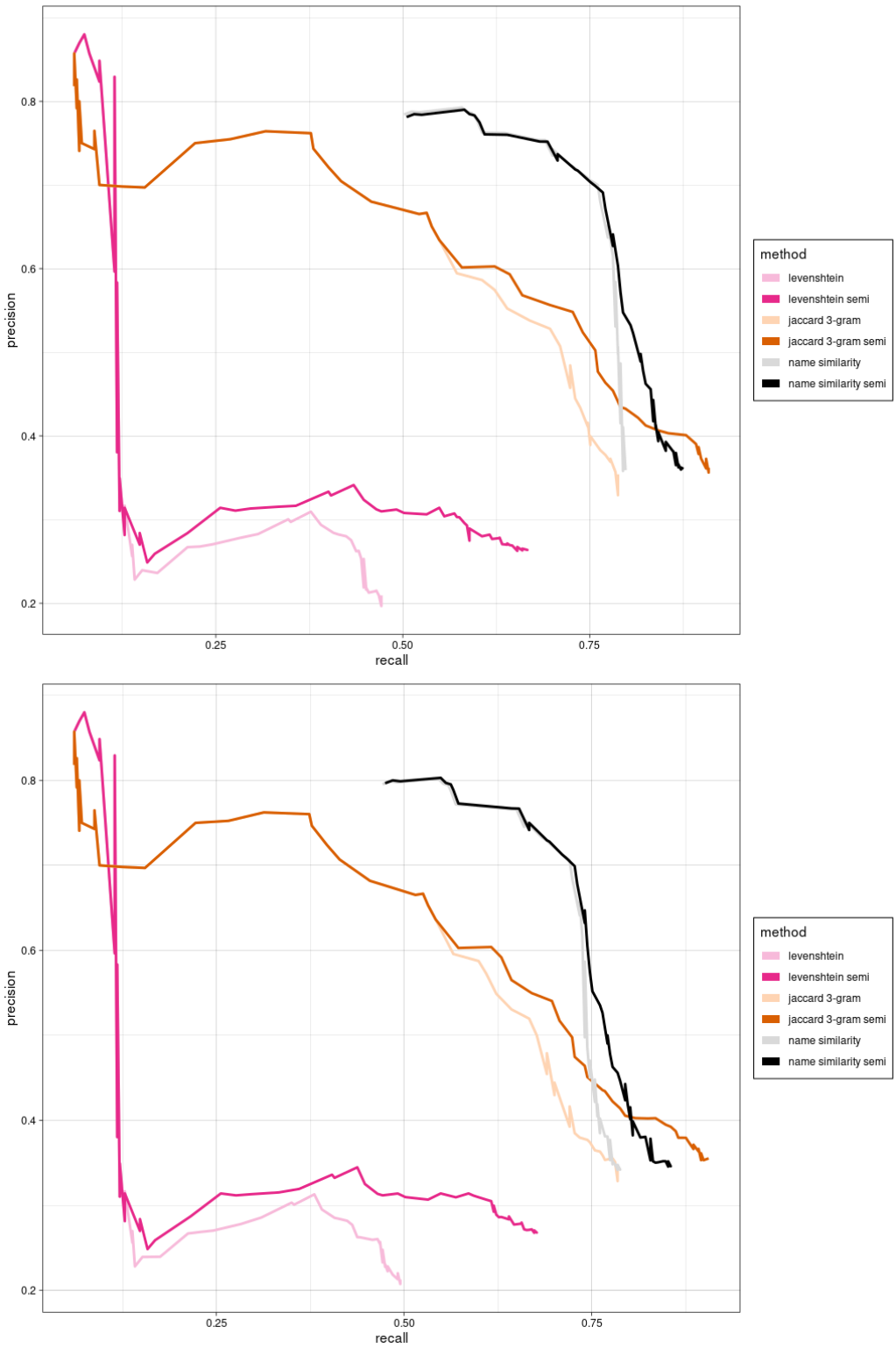


Figure 12: Relative performance of features on prefiltered (top) and unfiltered (bottom) data from the Colombia fieldsite.

The name similarity function was run with a Jaro-Winkler core function. Each model is represented twice: once in automated mode, and once in semi-automated mode.

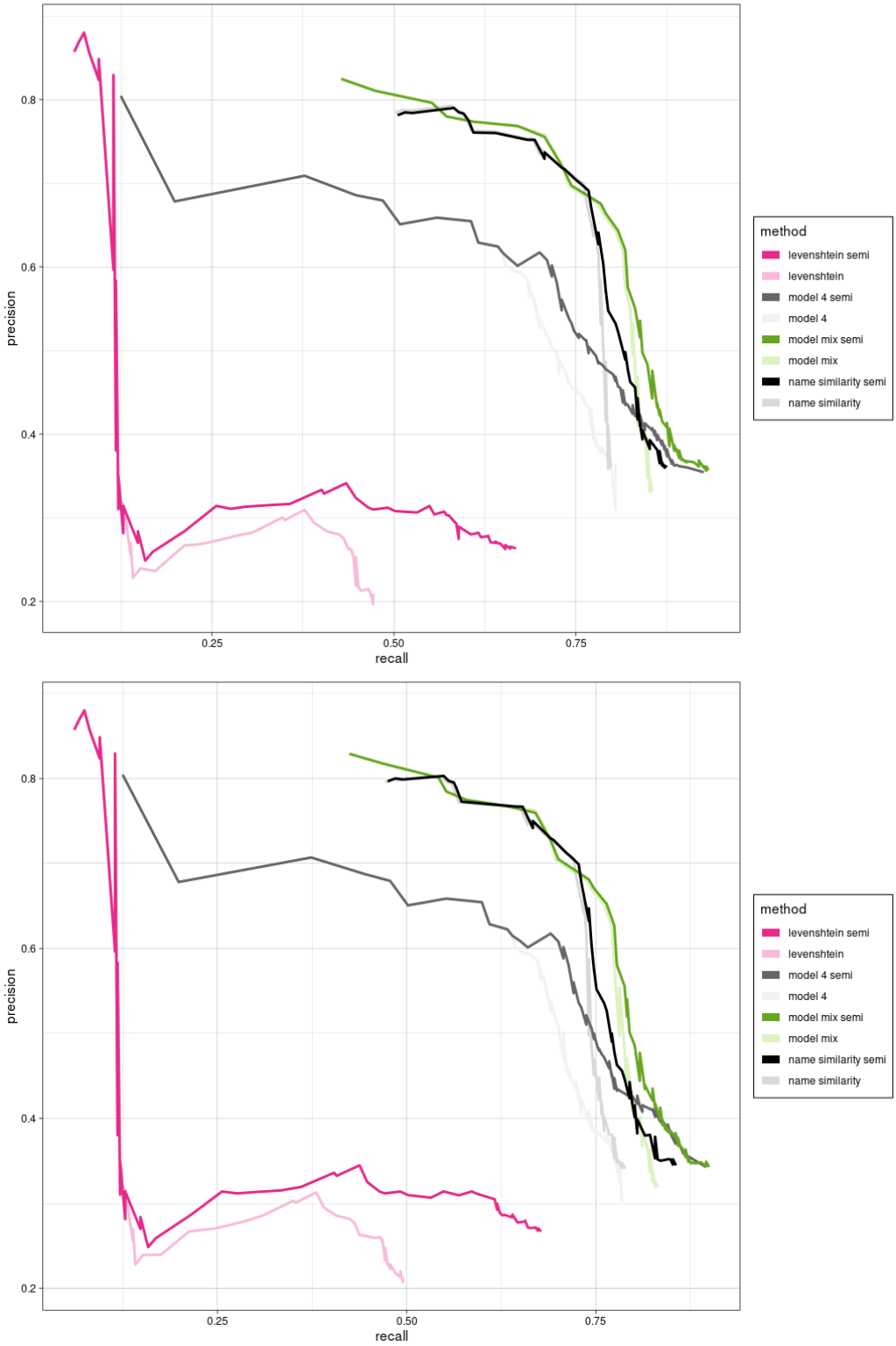


Figure 13: Relative performance of neural network models on prefiltered (top) and unfiltered (bottom) data from the Colombia fieldsite.

The name similarity function was run with a Jaro-Winkler core function. Each model is represented twice: once in automated mode, and once in semi-automated mode.

6 Discussion

Manual data collection introduces noise, particularly in name records where idiosyncratic ways of data recording are used. Such noisy name records may be a leading cause of errors in downstream analysis regimes, which leads to reduced usability and reproducibility of research.

In a scientific community where concerns are raised about the reproducibility of published research, standardized and automated data analysis methods can play an important role to raise standards. Methods to automate the record linkage and de-duplication of noisy strings have been developed for over a century, but existing string matching algorithms are not tailored to name records (the one exception being the Jaro-Winkler algorithm, but this method is only optimized for single names).

We have developed a new string similarity function that is tailored to name matching. Our new function is able to take human writing patterns of abbreviating or leaving out names into account, and works with records containing up to eight words per name[21]. By using the predictions of this function in conjunction with other string similarity metrics, we were able to train a neural network to make more accurate predictions than existing methods, with on average 10% improved recall and 15% improved precision over Levenshtein matching, as measured on simulated data.

On empirical data, we have found that Levenshtein matching does not even perform better than other common string matching methods. Our method has a 400% improved recall and 300% improved precision when compared with Levenshtein matching on the empirical data set. Compared to the best performing common string matching method, the 3-gram Jaccard similarity, our method has on average 20% improved recall and 25% improved precision.

We have observed that the neural network model does not always outperform the stand-alone name similarity function in precision, regardless of the extra available information. The neural network model does outperform the name similarity function in high recall regions (low threshold filtering), which is an important measure for the effectiveness of semi-automated matching.

In practice, while this new model may not always be accurate enough to perform fully automated name matching, it does reduce the amount of time that is needed by facilitating semi-automated name matching. Where existing methods provide only 50-75% recall when applied to empirical data, our model achieves up to 85% recall, reducing the amount of records that have to be matched manually.

6.1 Limitations and Future Directions

Even though our model performs better than common string matching methods, we will not achieve perfect classification. The nature of this problem prevents it–some noisy name records are simply too damaged to be recognized by humans or machines. Improved record linkage can be achieved by introducing more identifying information, such as age, gender, or ethnicity.

Furthermore, while our model only takes aleatoric uncertainty into account by calculating the probability of a match based on a subset of features, it does not consider epistemic uncertainty. That is, in this case, uncertainty that arises due to insufficient training data or training data that does not cover all parts of the trained feature space. Redesigning the neural network to be Bayesian could help to provide more transparent information about how certain the model is of its predictions. Methods for Bayesian neural networks have been described that can adapt the classical backpropagation algorithm to incorporate bayesian inference [33]. A Bayesian neural network uses probability distributions for the weights and biases, instead of fixed parameters. This property allows it to make predictions as if we trained and ran multiple regular neural network models with different initialization parameters. Research has also shown the usefulness of Bayesian methods in record linkage across arbitrarily many different documents, using multiple identifying parameters. The combination of such methods with our improved name matching model may increase accuracy even further.

[21] The limitation of eight words per name can be expanded as needed.

Bibliography

[1] Steven Minton and Claude Nanjo. A heterogeneous field matching method for record linkage. pages 314–321, 2005. doi: 10.1109/ICDM.2005.7.

[2] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data Quality and Record Linkage Techniques*. Springer Science & Business Media, 2007.

[3] Rebecca C. Steorts, Rob Hall, and Stephen E. Fienberg. A bayesian approach to graphical record linkage and de-duplication. 2015.

[4] Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. What does research reproducibility mean? *Science Translational Medicine*, 8(341):341ps12, 2016. doi: 10.1126/scitranslmed.aaf5027.

[5] Marcus R. Munafò, Brian A. Nosek, Doroty V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie Du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):1–9, 2017.

[6] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, 2016. doi: 10.1038/533452a.

[7] Rose Wiles, Gabriele Durrant, Sofie De Broe, and Jackie Powell. Assessment of needs for training in research methods in the uk social science community. Technical report, 2005.

[8] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, and Philip E. Bourne. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):1–9, 2016. doi: 10.1038/sdata.2016.18.

[9] Sheeba Samuel, Frank Löffler, and Birgitta König-Ries. Machine learning pipelines: Provenance, reproducibility and fair data principles. *Computing Research Repository*, abs/2006.12117, 2020.

[10] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, M. Kuhn, T. Lin Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686.

[11] Mark P. J. Van der Loo. The stringdist package for approximate string matching. *The R Journal*, 6(1): 111–122, 2014.

[12] Donald Hedeker and Robert D. Gibbons. *Longitudinal Data Analysis*, volume 451. John Wiley & Sons, 2006.

[13] Peter J. Carrington, John Scott, and Stan Wasserman. *Models and Methods in Social Network Analysis*, volume 28. Cambridge University Press, 2005.

[14] Paul E. Black. Levenshtein distance, 1999. URL https://www.nist.gov/dads/HTML/Levenshtein.html. Accessed 13 August 2021.

[15] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, pages 73–78. AAAI Press, 2003.

[16] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. doi: 10.1145/363958.363994.

[17] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29 (2):147–160, 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x.

[18] William E. Winkler. Overview of record linkage and current research directions. Technical report, U.S. Bureau of the Census, 2006.

[19] Peter Christen. A comparison of personal name matching: Techniques and practical issues. *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 290–294, 2006. doi: 10.1109/ICDMW.2006.2.

[20] Lawrence Philips. The double metaphone search algorithm. *C/C++ Users Journal*, 18(6):38–43, 2000.

[21] Alvaro E. Monge and Charles P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 267–270. AAAI Press, 1996.

[22] Martin Anthony and Peter L. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[23] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016.

[24] Sewall Wright. Systems of mating. v. general considerations. *Genetics*, 6(2):167–178, 1921. doi: 10.1093/genetics/6.2.167.

[25] Kenji Itao and Kunihiko Kaneko. Evolution of kinship structures driven by marriage tie and competition. *Proceedings of the National Academy of Sciences*, 7(3):2378–2384, 2020. doi: 10.1073/pnas.1917716117.

[26] Uwe D. Reichel. Language-independent grapheme-phoneme conversion and word stress assignment as a web service. In R. Hoffmann, editor, *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2014*, volume 71, pages 42–49. TUDpress, 2014.

[27] James Le. The top 10 machine learning algorithms every beginner should know. URL https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies.

[28] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

[29] Daniel Falbel and Javier Luraschi. *Torch for R*, 2021. URL https://github.com/mlverse/torch/. R package version 0.5.0.

[30] Jeremy Koster, Dieter Lukas, David Nolin, Eleanor Power, Alexandra Alvergne, Ruth Mace, Cody T. Ross, Karen Kramer, Russell Greaves, Mark Caudell, Shane MacFarlan, Eric Schniter, Robert Quinlan, Siobhan Mattison, Adam Reynolds, Chun Yi-Sum, and Eric Massengill. Kinship ties across the lifespan in human communities. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 374(1780), 2019. doi: 10.1098/rstb.2018.0069. Page 3, Table 1.

[31] Peter V Marsden. Network data and measurement. *Annual review of sociology*, 16(1):435–463, 1990.

[32] Cody T Ross and Daniel Redhead. Dietryin: An r package for data collection, automated data entry, and post-processing of network-structured economic games, social networks, and other roster-based dyadic data. *Behavior Research Methods*, pages 1–21, 2021.

[33] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

Appendix A Precision and Recall

In classification problems, precision and recall are two performance metrics that apply to data retrieval. To understand them, start by considering a confusion matrix that shows how many true positives, true negatives, false positives and false positives have been predicted by a classification model. For an example, look at figure 15.

Precision is now defined as the fraction of predicted positives that are true positives. In other words, how reliable is the positive prediction that the model has made.

Recall is defined as the fraction of true positives that have been correctly predicted to be true. Or, how many of the true positives were retrieved.

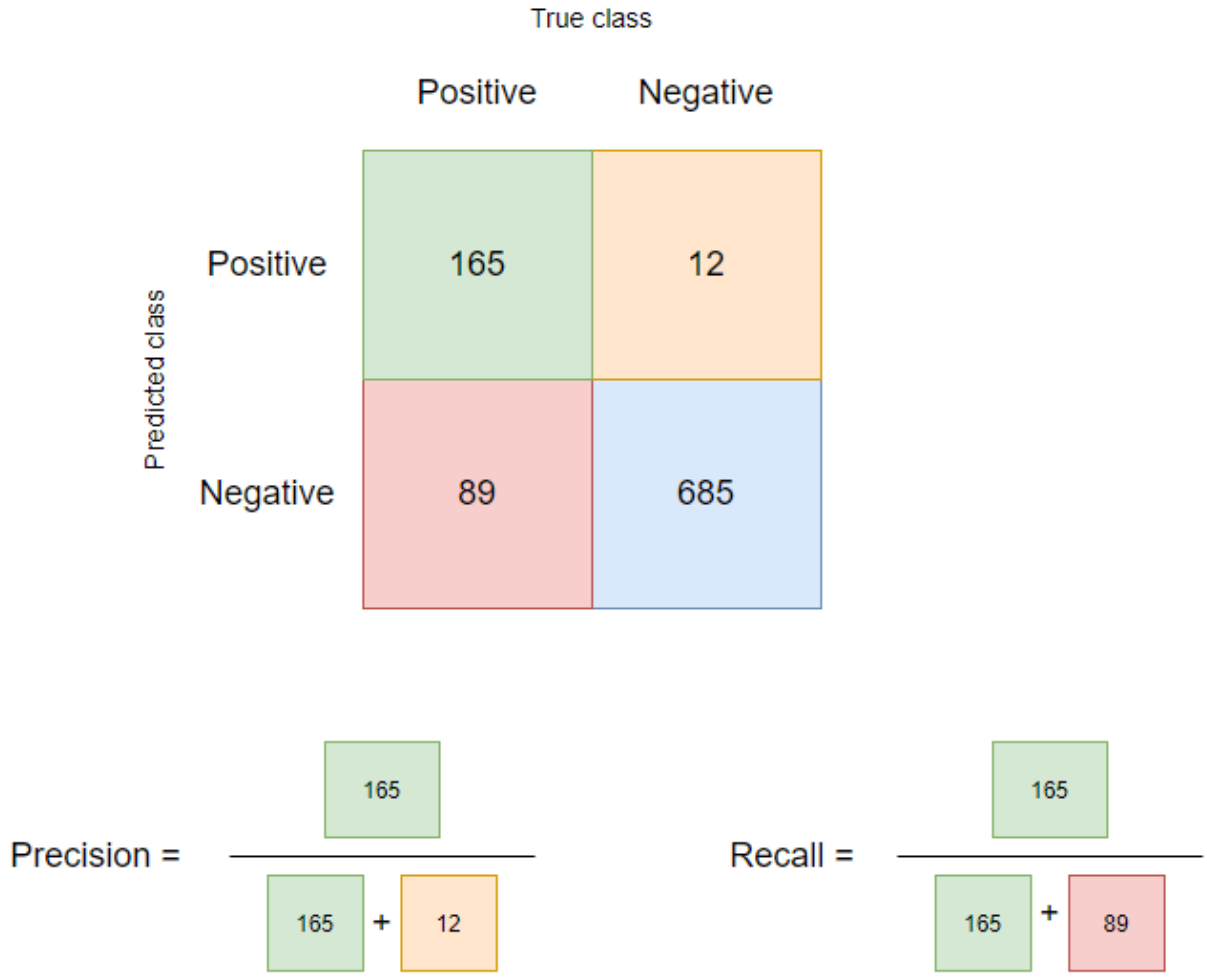


Figure 14: A confusion matrix.

Appendix B Full Feature Pairs Plot

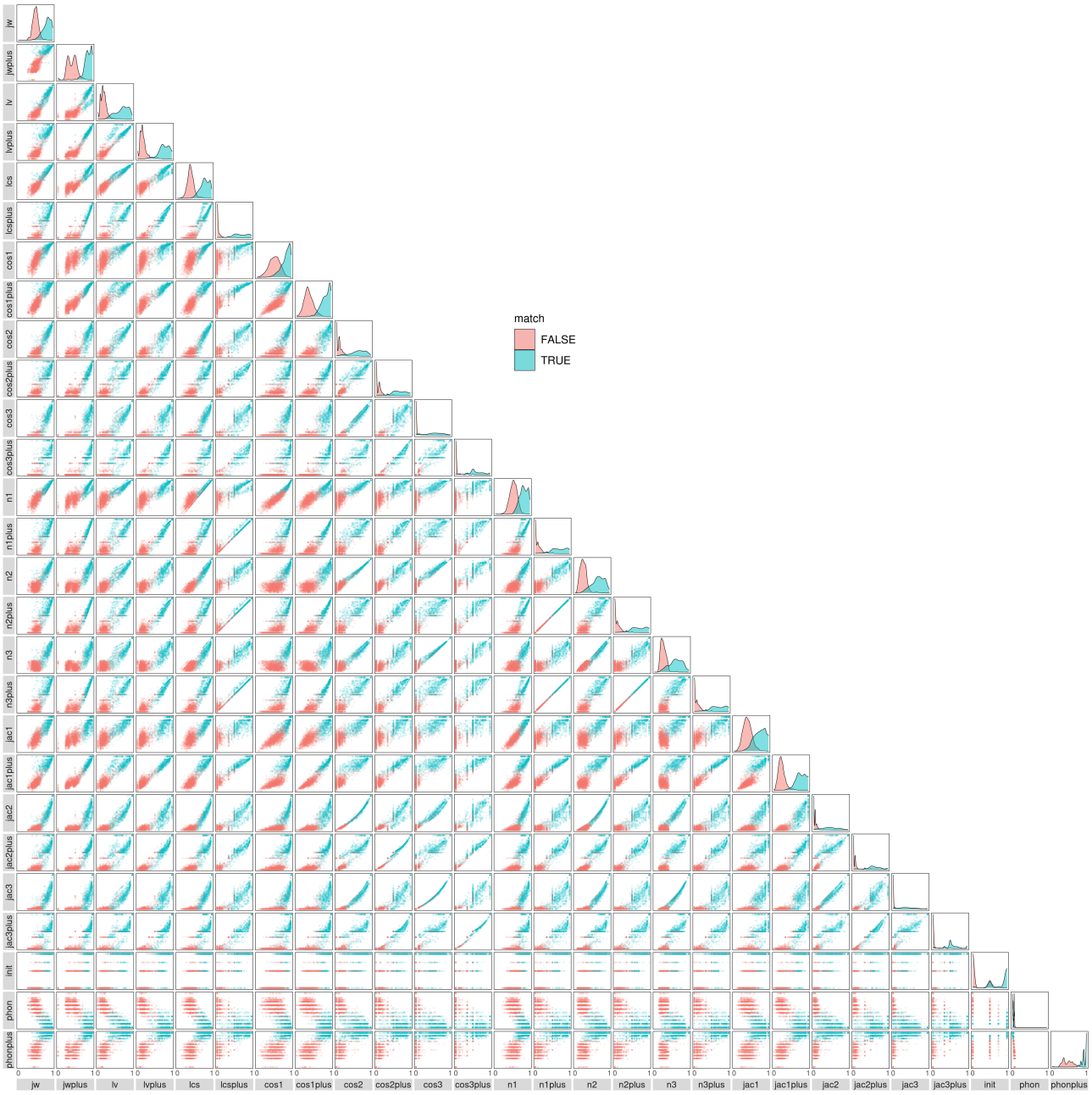


Figure 15: All pair plots for the features we have tested.

Appendix C Population Simulation Algorithm Code

```
# Declare the function. All the NULL parameters are there for customization by the end-user.
simulate_people_table <- function(n,
                                  seed = as.integer(format(Sys.time(), "%s")),
                                  # template activates a pre-determined list of parameters hard-coded in the function.
                                  template = NULL,
                                  # init_generation_mean determines the mean used for the ages of the initial generation of people.
                                  init_generation_mean = NULL,
                                  # marriage_name sets a naming convention for spouses. currently 'none', 'male', 'female'.
                                  marriage_name = NULL,
                                  # marriage_amount is a multiplier between 0 - 1 to indicate what fraction of the population is in a relationship.
                                  marriage_amount = NULL,
                                  # child_name sets a naming convention for children. currently 'paternal', 'maternal', 'paternal_maternal' and 'maternal_paternal'.
                                  child_name = NULL,
                                  # child_mean determines the mean used for the amount of children in a family.
                                  child_mean = NULL,
                                  # relationship coefficient
                                  rel_coef = 0.25) {

  # the current year will be used at the end of the script to calculate birthyears for everyone in the people table.
  current_year <- as.integer(format(Sys.Date(), "%Y"))

  # import a database of names with gender and ethnicity information from the United States Census bureau.
  load("./data/census_usa.rda")

  # import a database of country templates that specify family structures.
  country_templates <- utils::read.csv("./data/country_templates.csv", stringsAsFactors = F, encoding = "UTF-8")

  # load template settings
  if (template %in% country_templates$country) {
    init_generation_mean <- country_templates$init_generation_mean[which(country_templates$country == template)]
    marriage_name <- country_templates$marriage_name[which(country_templates$country == template)]
    marriage_amount <- country_templates$marriage_amount[which(country_templates$country == template)]
    child_name <- country_templates$child_name[which(country_templates$country == template)]
    child_mean <- country_templates$child_mean[which(country_templates$country == template)]
  } else {
    init_generation_mean <- 30
    marriage_name <- "male"
    marriage_amount <- 0.65
    child_name <- "paternal"
    child_mean <- 2
  }

  # generate the initial list of random unrelated people
  people <- data.frame(
    individ = id_maker(n, seed = seed),
    firstname = sample(1:nrow(firstnames), n, prob = firstnames$freq),
    lastname = sample(1:nrow(lastnames), n, prob = lastnames$freq),

    birthyear = rpois(n, init_generation_mean),
    gender = rep(NA, n),
```

```
    ethnicity = rep(NA, n),
    mother = id_maker(n, reserved = id_maker(n, seed = seed), seed = seed),
    father = id_maker(n, reserved = id_maker(2 * n, seed = seed), seed = seed),
    stringsAsFactors = F
  )

  people$gender <- firstnames$gender[people$firstname]
  people$firstname <- firstnames$firstname[people$firstname]

  people$ethnicity <- lastnames$ethnicity[people$lastname]
  people$lastname <- lastnames$lastname[people$lastname]

  # generate the initial list of marriages
  marriages <- data.frame(
    spouse.x = sample(people$individ[which(people$gender == "male")], round(n * marriage_amount / 2, 0)),
    spouse.y = sample(people$individ[which(people$gender == "female")], round(n * marriage_amount / 2, 0)),
    children = rep(0, round(n * marriage_amount / 2, 0)),
    children_prob = rep(0, round(n * marriage_amount / 2, 0)),
    stringsAsFactors = F
  )

  # generate a poisson distribution of number of children per family
  children_amount_distribution <- stats::rpois(max(round(n * marriage_amount / 2, 0), 100), child_mean)

  # generate a new column containing the probabilities that each family will produce another child
  for (i in 1:nrow(marriages)) {
    marriages$children_prob[i] <- length(which(children_amount_distribution > marriages$children[i])) / length(children_amount_distribution)
  }

  reserved_ids <- c(people$individ, people$mother, people$father)

  # initialize the single people vector for use in the main loop
  single_people <- c()

  m <- max(10, as.integer(n / 10))

  # generate the list of external singles
  external_people <- data.frame(
    individ = id_maker(m, seed = seed + 1, reserved = reserved_ids),
    firstname = sample(1:nrow(firstnames), m, prob = firstnames$freq),
    lastname = sample(1:nrow(lastnames), m, prob = lastnames$freq),

    birthyear = rpois(m, init_generation_mean) + 10,
    gender = rep(NA, m),
    ethnicity = rep(NA, m),
    mother = id_maker(m, reserved = c(reserved_ids, id_maker(m, seed = seed + 1)), seed = seed + 2),
    father = id_maker(m, reserved = c(reserved_ids, id_maker(2 * m, seed = seed + 1)), seed = seed + 2),
    stringsAsFactors = F
  )

  external_people$gender <- firstnames$gender[external_people$firstname]
  external_people$firstname <- firstnames$firstname[external_people$firstname]

  external_people$ethnicity <- lastnames$ethnicity[external_people$lastname]
  external_people$lastname <- lastnames$lastname[external_people$lastname]

  reserved_ids <- c(reserved_ids, external_people$individ, external_people$mother, external_people$father)

```

```
  # generate the actual people table
  for (i in 1:n) {
    individ <- id_maker(1, reserved = c(reserved_ids), seed = seed + i)

    reserved_ids <- c(reserved_ids, individ)
    # select a mother from the marriage table, based on probability that she and her spouse will
        have a new kid
    mother <- sample(marriages$spouse.y, 1, prob = marriages$children_prob)
    father <- marriages$spouse.x[which(marriages$spouse.y == mother)]

    # add one child to the child count of this couple
    marriages$children[which(marriages$spouse.x == father & marriages$spouse.y == mother)] <-
        marriages$children[which(marriages$spouse.x == father & marriages$spouse.y == mother)] + 1

    # define how the last name of the child is generated
    if (child_name == "paternal") {
      lastname <- people$lastname[which(people$individ == father)]
    } else if (child_name == "maternal") {
      lastname <- people$lastname[which(people$individ == mother)]
    } else if (child_name == "paternal_maternal") {
      # copy the first word of the father's surname and the first word of the mother's surname
      lastname <- paste(gsub("([A-Za-z]+).*", "\\1", people$lastname[which(people$individ == father)
          ]), gsub("([A-Za-z]+).*", "\\1", people$lastname[which(people$individ == mother)]))
    } else if (child_name == "maternal_paternal") {
      # copy the first word of the mother's surname and the first word of the father's surname
      lastname <- paste(gsub("([A-Za-z]+).*", "\\1", people$lastname[which(people$individ == mother)
          ]), gsub("([A-Za-z]+).*", "\\1", people$lastname[which(people$individ == father)]))
    }

    # determine the ethnicity of the father
    if(father %in% people$individ){
      father_ethn <- people$ethnicity[people$individ == father]
      father_ethn <- father_ethn
    } else if(father %in% external_people$individ){
      father_ethn <- external_people$ethnicity[external_people$individ == father]
      father_ethn <- father_ethn
    } else {
      father_ethn <- "mix"
    }

    # determine the ethnicity of the mother
    if(mother %in% people$individ){
      mother_ethn <- people$ethnicity[people$individ == mother]
      mother_ethn <- mother_ethn
    } else if(mother %in% external_people$individ){
      mother_ethn <- external_people$ethnicity[external_people$individ == mother]
      mother_ethn <- mother_ethn
    } else {
      mother_ethn <- "mix"
    }

    # set the ethnicity of the child
    if(mother_ethn == father_ethn){
      ethnicity <- mother_ethn
    } else {
      ethnicity <- "mix"
    }

    # generate a firstname and gender
    firstname_id <- sample(1:nrow(firstnames), 1, prob = firstnames$freq)
    firstname <- firstnames$firstname[firstname_id]
    gender <- firstnames$gender[firstname_id]

    # generate a birthyear that is at least 18 years later than either father or mother
```

```
    birthyear <- sample(rpois(100, init_generation_mean - 3), 1) + max(people$birthyear[which(people
        $individ %in% c(father, mother))])

    # this if-statement determines if this child will ever enter a relationship
    if (sample(c(TRUE, FALSE), 1, prob = c(marriage_amount, 1 - (marriage_amount)))) {
      # select all available single people from the opposite sex
      single_people <- c(single_people, individ, external_people$individ)
      singles <- people[which(people$individ %in% single_people & people$gender != gender & abs(
          people$birthyear - birthyear) < 20), ]
      singles_external <- external_people[which(external_people$gender != gender & abs(external_
          people$birthyear - birthyear) < 20), ]
      singles <- rbind(singles, singles_external)

      # remove everybody who is related - incest taboo
      if (nrow(singles) != 0){
        kinship <- rbind(people[, c("individ", "father", "mother")], data.frame(individ = individ,
            father = father, mother = mother, stringsAsFactors = F))
        for(m in 1:nrow(singles)){
          if(relatedness(individ, singles$individ[m], kinship) > rel_coef){
            singles <- singles[-m, ]
          }
        }
      }

      if (nrow(singles) != 0) {
        # select a partner. partners with a large age difference are less likely to be chosen.
        partner <- sample(singles$individ, 1, prob = (1 / (abs(birthyear - singles$birthyear)**2 +
            0.01)))

        # generate a new entry for the marriage table
        if (gender == "male") {
          marriage_add <- data.frame(spouse.x = individ, spouse.y = partner, children = 0, children_
              prob = 0, stringsAsFactors = F)
        } else if (gender == "female") {
          marriage_add <- data.frame(spouse.x = partner, spouse.y = individ, children = 0, children_
              prob = 0, stringsAsFactors = F)
        }
        marriages <- rbind(marriages, marriage_add)

        # recalculate the probability of each family to have a new child
        for (j in 1:nrow(marriages)) {
          marriages$children_prob[j] <- length(which(children_amount_distribution > marriages$
              children[j])) / length(children_amount_distribution)
        }

        # refresh the single people list so that all married people are removed
        single_people <- single_people[single_people %!in% c(marriages$spouse.x, marriages$spouse.y)
            ]
        external_people <- external_people[external_people$individ %!in% c(marriages$spouse.x,
            marriages$spouse.y),]

        # add a new external person to the external people table
        add_external <- data.frame(
          individ = id_maker(1, seed = seed + 1, reserved = reserved_ids),
          firstname = sample(1:nrow(firstnames), 1, prob = firstnames$freq),
          lastname = sample(1:nrow(lastnames), 1, prob = lastnames$freq),

          birthyear = birthyear + sample(5:10, 1),
          gender = rep(NA, 1),
          ethnicity = rep(NA, 1),
          mother = id_maker(1, reserved = reserved_ids, seed = seed + i + 1),
          father = id_maker(1, reserved = reserved_ids, seed = seed + i + 2),
          stringsAsFactors = F
        )
```

```
223         reserved_ids <- c(reserved_ids, add_external$individ, add_external$mother, add_external$
                father)
224
225         add_external$gender <- firstnames$gender[add_external$firstname]
226         add_external$firstname <- firstnames$firstname[add_external$firstname]
227
228         add_external$ethnicity <- lastnames$ethnicity[add_external$lastname]
229         add_external$lastname <- lastnames$lastname[add_external$lastname]
230
231         external_people <- rbind(external_people, add_external)
232
233       }
234     }
235
236     # generate a new entry for the people table
237     people_add <- data.frame(
238       individ = individ,
239       firstname = firstname,
240       lastname = lastname,
241       birthyear = birthyear,
242       gender = gender,
243       ethnicity = ethnicity,
244       mother = mother,
245       father = father,
246       stringsAsFactors = F
247     )
248
249     reserved_ids <- c(reserved_ids, individ)
250
251     people <- rbind(people, people_add)
252   }
253
254   # move all birthyears forward so that the most recent person is born this year
255   people$birthyear <- people$birthyear + current_year - max(people$birthyear)
256
257   # print the amount of people who have children per generation
258   cat(paste0("fraction of parents per generation = ", round(nrow(people[people$individ %in% c(
          marriages$spouse.x, marriages$spouse.y), ]) / nrow(people), 2), "\n"))
259
260   # create a kinship table
261   kinship_table <- data.frame(individ = people$individ, individ_age = people$birthyear, mother =
          people$mother, father = people$father, stringsAsFactors = F)
262   kinship_table$mother_age <- left_join(data.frame(individ = kinship_table$mother, stringsAsFactors
          = F), people[,c("individ", "birthyear")], by = "individ")[,"birthyear"]
263   kinship_table$father_age <- left_join(data.frame(individ = kinship_table$father, stringsAsFactors
          = F), people[,c("individ", "birthyear")], by = "individ")[,"birthyear"]
264   kinship_table$mother_agedif <- kinship_table$individ_age - kinship_table$mother_age
265   kinship_table$father_agedif <- kinship_table$individ_age - kinship_table$father_age
266   # print the mean age difference between generations
267   cat(paste0("mean generation spread = ", round(mean(c(kinship_table$mother_agedif, kinship_table$
          father_agedif), na.rm = T), 2), "\n"))
268
269   # add birthyear information to the marriage table
270   spouse_age <- people[, c("individ", "birthyear")]
271   colnames(spouse_age)[1] <- "spouse.x"
272   marriages$birthyear.x <- left_join(marriages[, "spouse.x", drop = F], spouse_age, by = "spouse.x")
          [, 2]
273   colnames(spouse_age)[1] <- "spouse.y"
274   marriages$birthyear.y <- left_join(marriages[, "spouse.y", drop = F], spouse_age, by = "spouse.y")
          [, 2]
275
276   # print the mean number of children per family
277   cat(paste0("mean children per family = ", round(mean(marriages$children), 2), "\n"))
278   # return the people table without the initial list of random people
```

```
  return(people[(n + 1):(2 * n), ])
}


# END OF SCRIPT
```

Appendix D Misspelling Generator Code

```
# x = name spelled in latin characters
# y = name spelled in IPA (international phonetic alphabet)
# abbr = probability that one of the names gets abbreviated, (first, middle, last)
# miss = probability that one of the names gets deleted, (first, middle, last)
# phon = probability that one of the names is phonetically spelled, (first, middle, last)
misspell <- function(x, y, abbr = c(0.3, 0.1, 0.2), miss = c(0.1,0.6,0.1), phon = c(0.9,0.7,0.5),
    language = "english"){

  x <- as.character(x)
  y <- as.character(y)

  x <- unlist(strsplit(x, " "))
  phonetic <- unlist(strsplit(y, " "))

  if(length(x) == 1){

    abbr <- abbr[1]
    miss <- miss[1]
    phon <- phon[1]

    if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr, 1 - abbr))){
      x <- substring(x, 1, 1)
    }

    if(nchar(x) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(miss, 1 - miss))){
      x <- ""
    }

    if(nchar(x) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon, 1 - phon))){
      x <- phonetic_write(phonetic, language = language)
    }

    if(nchar(x) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon, 1 - phon))){
      x <- edit_distance(x)
    }

  } else if(length(x) == 2){
    if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr[1], 1 - abbr[1]))){
      x[1] <- substring(x[1], 1, 1)
    } else if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr[length(abbr)], 1 - abbr[length(abbr)]))
        ){
      x[2] <- substring(x[2], 1, 1)
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(miss[1], 1 - miss[1]))){
      x[1] <- ""
    } else if(nchar(x[2]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(miss[length(miss)], 1 -
        miss[length(miss)]))){
      x[2] <- ""
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[1], 1 - phon[1]))){
      x[1] <- phonetic_write(phonetic[1], language = language)
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[length(phon)], 1 - phon[
        length(phon)]))){
      x[1] <- edit_distance(x[1])
    }

    if(nchar(x[2]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[length(phon)], 1 - phon[
        length(phon)]))){
```

```
      x[2] <- phonetic_write(phonetic[2], language = language)
    }

    if(nchar(x[2]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[length(phon)], 1 - phon[
        length(phon)]))){
      x[2] <- edit_distance(x[2])
    }

    x <- x[!is.na(x) & x != ""]
    x <- paste(x, collapse = " ")

  } else if(length(x) > 2){
    if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr[1], 1 - abbr[1]))){
      x[1] <- substring(x[1], 1, 1)
    } else if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr[2], 1 - abbr[2]))){
      x[2:(length(x)-1)] <- substring(x[2:(length(x)-1)], 1, 1)
    } else if(sample(c(TRUE, FALSE), size = 1, prob = c(abbr[3], 1 - abbr[3]))){
      x[length(x)] <- substring(x[length(x)], 1, 1)
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(miss[1], 1 - miss[1]))){
      x[1] <- ""
    } else if(sample(c(TRUE, FALSE), size = 1, prob = c(miss[2], 1 - miss[2]))){
      x[2:(length(x)-1)] <- ""
    } else if(nchar(x[length(x)]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(miss[3], 1 - miss
        [3]))){
      x[length(x)] <- ""
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[1], 1 - phon[1]))){
      x[1] <- phonetic_write(phonetic[1], language = language)
    }

    if(nchar(x[1]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[1], 1 - phon[1]))){
      x[1] <- edit_distance(x[1])
    }

    if(sample(c(TRUE, FALSE), size = 1, prob = c(phon[2], 1 - phon[2]))){
      x[2:(length(x)-1)] <- phonetic_write(phonetic[x[2:(length(x)-1)]], language = language)
    }

    if(sample(c(TRUE, FALSE), size = 1, prob = c(phon[2], 1 - phon[2]))){
      for(p in 2:(length(x)-1))
      x[p] <- edit_distance(x[p])
    }

    if(nchar(x[length(x)]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[3], 1 - phon[3]))){
      x[length(x)] <- phonetic_write(phonetic[length(x)], language = language)
    }

    if(nchar(x[length(x)]) != 1 & sample(c(TRUE, FALSE), size = 1, prob = c(phon[3], 1 - phon[3]))){
      x[length(x)] <- edit_distance(x[length(x)])
    }

    x <- x[!is.na(x) & x != ""]
    x <- paste(x, collapse = " ")
  }

  return(x)
}

# This function attempts to write the input string in a phonetic way
phonetic_write <- function(x, language = "english"){

```

```
  x <- as.character(x)
  load("./data/ipa_reverse_enzo.rda")
  ipa_reverse <- ipa_reverse[order(nchar(ipa_reverse[,1]), decreasing = T),]

  for(i in 1:nrow(ipa_reverse)){
    if(ipa_reverse[i,language] != ""){
      replace <- sample(unlist(strsplit(ipa_reverse[i,language], "\\.")), 1)
      x <- gsub(ipa_reverse[i,1], replace, x)
    } else {
      x <- gsub(ipa_reverse[i,1], "", x)
    }
  }
  return(x)
}

# ins = probability for insertion of random character
# del = probability of deleting a random character from the string
# sub = probability of subsituting a random character from the string by a random other character
# tra = probability of transposing two characters
edit_distance <- function(x, ins = 0.005, del = 0.05, sub = 0.001, tra = 0){

  x <- as.character(x)

  if(is.na(x) | is.null(x) | length(x) < 1){
    return("")
  }

  if(nchar(x) < 2){
    return(x)
  }

  x <- unlist(strsplit(x, ""))

  for(i in 2:length(x)){

    if(sample(c(TRUE, FALSE), size = 1, prob = c(ins, 1 - ins))){
      x <- c(x[1:(i-1)], sample(letters, 1), x[i:length(x)])
    }

    if(sample(c(TRUE, FALSE), size = 1, prob = c(del, 1 - del))){
      x <- x[-i]
    }

    if(sample(c(TRUE, FALSE), size = 1, prob = c(sub, 1 - sub))){
      x <- x[-i]
      x <- c(x[1:(i-1)], sample(letters, 1), x[i:length(x)])
    }

    if(sample(c(TRUE, FALSE), size = 1, prob = c(tra, 1 - tra))){
      t1 <- x[i-1]
      t2 <- x[i]
      x[i-1] <- t2
      x[i] <- t1
    }
  }
  x <- x[!is.na(x)]
  x <- paste0(x, collapse = "")
  return(x)
}


# END OF SCRIPT
```

Appendix E Name Similarity Algorithm Code

```
string.compare <- function(Var1, Var2, method = "jw", q = 1) {

  # the line below this one is for debugging purposes.
  # cat(paste(Var2, "and", Var1, "\n"))
  if(is.na(Var1) | is.na(Var2)){
    return(0)
  }
  if(nchar(Var1) == 0 | nchar(Var2) == 0){
    return(0)
  }
  # clean up both strings and split the individual words.
  string1 <- Var1 %>%
    tolower() %>%
    trimws() %>%
    str_replace_all(pattern = "[[:punct:]]", replacement = " ") %>%
    strsplit(" ") %>%
    unlist()
  string2 <- Var2 %>%
    tolower() %>%
    trimws() %>%
    str_replace_all(pattern = "[[:punct:]]", replacement = " ") %>%
    strsplit(" ") %>%
    unlist()

  # create a matrix whose dimensions are the number of words in each string.
  compare <- array(NA, dim = c(
    length(string1),
    length(string2)
  ), dimnames = list(
    string1,
    string2
  ))

  # assign a match score between each set of words.
  compare[] <- do.call(
    mapply,
    c(
      list(FUN = string.score, method = method, q = q),
      expand.grid(dimnames(compare), stringsAsFactors = FALSE)
    )
  )

  # determine which allowed order of matching words gives the highest total match score.
  max.sum <- string.max(compare)

  if(is.numeric(max.sum) & !is.nan(max.sum) & !is.na(max.sum) & !is.null(max.sum)){
    if (max.sum > 0) {

      # normalize the match score.
      score <- round(max.sum / sum(sapply(string1, function(x) string.score(x, x))) * 100, 3)
      return(score)
    } else {
      return(0)
    }
  } else {
    return(0)
  }

}

name_similarity <- Vectorize(string.compare)

```

```
63
64 # The string.max function determines which allowed order of matching the words
65 # from two strings together gives the highest total match score.
66 # input is an mxn-matrix.
67
68 string.max <- function(input) {
69   if (nrow(input) == 0 | ncol(input) == 0) {
70     return(0)
71   } else if (nrow(input) == 1 | ncol(input) == 1) {
72     return(max(input))
73   }
74
75   permutations <- readRDS("./data/permutations.RDS")
76
77   permutations <- substr(permutations, 1, nrow(input))
78   permutations <- permutations[!grepl(paste0(c((ncol(input) + 1):9), collapse = "|"), permutations)]
79   permutations <- unique(permutations)
80
81   sums <- 0
82   for (i in 1:length(permutations)) {
83     add <- 0
84     for (j in 1:nrow(input)) {
85       if (!is.null(as.numeric(substr(permutations[i], j, j))) & !is.na(as.numeric(substr(
             permutations[i], j, j)))) {
86         if (as.numeric(substr(permutations[i], j, j)) > 0) {
87           add <- add + input[j, as.numeric(substr(permutations[i], j, j))]
88         }
89       }
90     }
91     if(!is.nan(add) & !is.null(add) & !is.na(add) & is.numeric(add) & length(add) == 1){
92       sums <- max(c(sums, add))
93     }
94   }
95
96   if(is.nan(sums) | is.na(sums) | is.null(sums) | !is.numeric(sums) | sums == 0){
97     return(0)
98   } else if (length(sums) > 0 & any(sums > 0)) {
99     return(max(sums))
100  } else {
101    return(0)
102  }
103 }
104
105
106 # The string.score function assigns a matching score to two words.
107 # Var1 is the first word.
108 # Var2 is the second word.
109
110 string.score <- function(Var1, Var2, method = "jw", q = 1) {
111
112   if (is.null(Var1) | is.null(Var2) | is.na(Var1) | is.na(Var2) | Var1 == "" | Var2 == "") { # if
          one of the entries is empty, score 0
113     return(0)
114   } else if (Var1 == substr(Var2, 1, 1)) { # if Var1 is an abbreviation of Var2, score 50
115     return(50)
116   } else if (nchar(Var1) == 1) { # if Var1 is an abbreviation but not of Var2, score 0
117     return(0)
118   } else if (Var2 == substr(Var1, 1, 1)) { # if Var2 is an abbreviation of Var1, score 50
119     return(50)
120   } else if(nchar(Var2) == 1) {
121     return(0)
122   } else { # give a score based on the 1-gram cosine string distance
123     if(method %in% c("jw", "cos", "jaccard")){
124       return(round(100 - (100 * stringdist(Var1, Var2, method = method, q = q)), 3))
```

```
125     } else {
126       return(round(100 - (100 * stringdistnorm(Var1, Var2, method = method)), 3))
127     }
128   }
129 }
130
131
132 # END OF SCRIPT
```

Appendix F Neural Network Training Code

```
library(torch)

ann <- function(x, y, neurons = 16, iterations = 1000, learning_rate = 1e-1, mode = "reg", model =
    NULL, ndropout = 0.1, device = "cpu"){

  x <- as.matrix(x)
  y <- as.matrix(as.numeric(y))

  d_in <- ncol(x)
  d_out <- ncol(y)
  d_hidden <- neurons

  x <- torch_tensor(x)
  y <- torch_tensor(y)

  # mean values of the neurons in the first and second layers
  if(is.null(model)){
    w1 <- torch_randn(d_in, d_hidden, requires_grad = TRUE, device = device)
    b1 <- torch_randn(1, d_hidden, requires_grad = TRUE, device = device)
    w2 <- torch_randn(d_hidden, d_hidden, requires_grad = TRUE, device = device)
    b2 <- torch_randn(1, d_hidden, requires_grad = TRUE, device = device)
    w3 <- torch_randn(d_hidden, d_out, requires_grad = TRUE, device = device)
  } else {
    w1 <- torch_tensor(model[[1]], requires_grad = TRUE, device = device)
    w2 <- torch_tensor(model[[2]], requires_grad = TRUE, device = device)
    w3 <- torch_tensor(model[[3]], requires_grad = TRUE, device = device)
    b1 <- torch_tensor(model[[4]], requires_grad = TRUE, device = device)
    b2 <- torch_tensor(model[[5]], requires_grad = TRUE, device = device)
  }

  timeStart <- Sys.time()
  loss_count <- c()
  loss_total <- c()

  for (t in 1:iterations) {

    # ### -------- Error --------
    # e_w1 <- torch_randn(d_in, d_hidden) / 100
    # e_w2 <- torch_randn(d_hidden, d_hidden) / 100
    # e_w3 <- torch_randn(d_hidden, d_out) / 100
    #
    # w1a <- w1 + e_w1
    # w2a <- w2 + e_w2
    # w3a <- w3 + e_w3

    ### -------- Dropout --------
    dropout <- nn_dropout(p = ndropout)
    w1d <- dropout(w1)
    w2d <- dropout(w2)
    w3d <- dropout(w3)

    ### -------- Forward pass --------
    y_hat <- torch_sigmoid(torch_tanh(torch_tanh(x$mm(w1d)$add(b1))$mm(w2d)$add(b2))$mm(w3d))

    ### -------- compute loss --------
    if(mode == "reg"){
      bce_loss <- nn_mse_loss()
      loss <- bce_loss(y_hat, y)
    } else if(mode == "cat"){
      loss <- nnf_binary_cross_entropy(y_hat, y)
    }

```

```
    # error catching
    if(is.nan(loss$item())){
      break("loss is nan")
    }
    # loss_count <- c(loss_count, loss$item())
    # loss_total <- c(loss_total, loss$item())
    # if (t %% 100 == 0){
    #   timeEnd <- Sys.time()
    #   cat("Epoch: ", t, "   Loss: ", mean(loss_count), "    time: ", difftime(timeEnd, timeStart,
        units='mins'), "\n")
    #   cat("Total Loss: ", mean(loss_total), "\n")
    #   loss_count <- c()
    # }

    ### -------- Backpropagation --------
    loss$backward()

    ### -------- Update weights --------

    # Wrap in with_no_grad() because this is a part we DON'T
    # want to record for automatic gradient computation
    with_no_grad({
      w1 <- w1$sub_(learning_rate * w1$grad)
      w2 <- w2$sub_(learning_rate * w2$grad)
      w3 <- w3$sub_(learning_rate * w3$grad)
      b1 <- b1$sub_(learning_rate * b1$grad)
      b2 <- b2$sub_(learning_rate * b2$grad)

      # Zero gradients after every pass
      w1$grad$zero_()
      b1$grad$zero_()
      b2$grad$zero_()
      w2$grad$zero_()
      w3$grad$zero_()
    })

  }

  return(list(matrix(as.numeric(w1), ncol = d_hidden),
              matrix(as.numeric(w2), ncol = d_hidden),
              matrix(as.numeric(w3), ncol = d_out),
              matrix(as.numeric(b1), ncol = d_hidden),
              matrix(as.numeric(b2), ncol = d_hidden)))
}


# END OF SCRIPT
```

Appendix G Neural Network Prediction Code

```
library(torch)

neg_pred <- function(x, model){

  x <- as.matrix(x)
  x <- torch_tensor(x)

  d_in <- dim(model[[1]])[1]
  d_hidden <- dim(model[[1]])[2]
  d_out <- 1

  w1 <- torch_tensor(model[[1]])
  w2 <- torch_tensor(model[[2]])
  w3 <- torch_tensor(model[[3]])
  b1 <- torch_tensor(model[[4]])
  b2 <- torch_tensor(model[[5]])

  y_hat <- torch_sigmoid(torch_tanh(torch_tanh(x$mm(w1)$add(b1))$mm(w2)$add(b2))$mm(w3))

  return(as.numeric(y_hat))
}


# END OF SCRIPT
```

Appendix H Blocking Code

```
1
2
3  # Clear workspace
4  rm(list = ls())
5
6  # Load functions
7  library(stringdist)
8  source("./R/functions.R")
9
10 # Load data
11 load("./test_data/testdata_simulated.rda")
12 # Select 1000 matches
13 matched_names <- matched_names[sample(1:nrow(matched_names), 1000),]
14
15 # Generate all dyads
16 data <- expand.grid(unique(matched_names$nomination), unique(matched_names$match), stringsAsFactors
       = F)
17 data$match <- 0
18
19 # Assign "1" to all dyads that represent a match
20 for(i in 1:nrow(matched_names)){
21   data$match[data$Var1 == matched_names$nomination[i] & data$Var2 == matched_names$match[i]] <- 1
22   if(i %% 10 == 0){
23     cat(paste0(i/1000, "\n"))
24   }
25 }
26
27 n_m <- 1000 # How many samples to take of the dyads?
28
29 data <- data[sample(1:nrow(data), n_m),]
30
31 data$Var1 -> nomination
32 data$Var2 -> reference
33
34 n_i <- 10  # how many iterations to determine the speed of each feature?
35 speed <- data.frame(feature = NA, speed = NA, stringsAsFactors = F)[0,]
36
37 # Generating advanced features JW ----
38 t <- 0
39 for(i in 1:n_i){
40   start <- Sys.time()
41   data$jw = 1 - stringdist(nomination, reference, method = "jw")
42   end <- Sys.time()
43   cat("JW \n")
44   t <- t + as.numeric(as.numeric(difftime(end, start, units = "secs")))
45 }
46 speed <- rbind(speed, data.frame(feature = "JW", speed = n_m * n_i / t, stringsAsFactors = F))
47
48 t <- 0
49 for(i in 1:n_i){
50   start <- Sys.time()
51   data$jwplus = string_compare(nomination, reference, method = "jw") / 100
52   end <- Sys.time()
53   cat("\n JW+ \n")
54   t <- t + as.numeric(difftime(end, start, units = "secs"))
55 }
56 speed <- rbind(speed, data.frame(feature = "JW+", speed = n_m * n_i / t, stringsAsFactors = F))
57 # Generating advanced features LV ----
58 t <- 0
59 for(i in 1:n_i){
60   start <- Sys.time()
61   data$lv = 1 - stringdistnorm(nomination, reference, method = "lv")
```

```
  end <- Sys.time()
  cat("\n LV \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "LV", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$lvplus = string_compare(nomination, reference, method = "lv") / 100
  end <- Sys.time()
  cat("\n LV+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "LV+", speed = n_m * n_i / t, stringsAsFactors = F))
# Generating advanced features LCS ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$lcs = 1 - stringdistnorm(nomination, reference, method = "lcs") / 2
  end <- Sys.time()
  cat("\n LCS \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "LCS", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$lcsplus = string_compare(nomination, reference, method = "lcs") / 100
  end <- Sys.time()
  cat("\n LCS+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "LCS+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 1-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos1 = 1 - stringdist(nomination, reference, method = "cos", q = 1)
  end <- Sys.time()
  cat("\n COS1 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS1", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos1plus = string_compare(nomination, reference, method = "cos", q = 1) / 100
  end <- Sys.time()
  cat("\n COS1+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS1+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 2-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos2 = 1 - stringdist(nomination, reference, method = "cos", q = 2)
  end <- Sys.time()
  cat("\n COS2 \n")
```

```
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS2", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos2plus = string_compare(nomination, reference, method = "cos", q = 2) / 100
  end <- Sys.time()
  cat("\n COS2+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS2+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 3-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos3 = 1 - stringdist(nomination, reference, method = "cos", q = 3)
  end <- Sys.time()
  cat("\n COS3 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS3", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$cos3plus = string_compare(nomination, reference, method = "cos", q = 3) / 100
  end <- Sys.time()
  cat("\n COS3+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "COS3+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 1-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n1 = 1 - stringdistnorm(nomination, reference, method = "qgram", q = 1) / 2
  end <- Sys.time()
  cat("\n N1 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "N1", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n1plus = string_compare(nomination, reference, method = "qgram", q = 1) / 100
  end <- Sys.time()
  cat("\n N1+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "N1+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 2-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n2 = 1 - stringdistnorm(nomination, reference, method = "qgram", q = 2) / 2
  end <- Sys.time()
  cat("\n N2 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
```

```
}
speed <- rbind(speed, data.frame(feature = "N2", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n2plus = string_compare(nomination, reference, method = "qgram", q = 2) / 100
  end <- Sys.time()
  cat("\n N2+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "N2+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 3-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n3 = 1 - stringdistnorm(nomination, reference, method = "qgram", q = 3) / 2
  end <- Sys.time()
  cat("\n N3 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "N3", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$n3plus = string_compare(nomination, reference, method = "qgram", q = 3) / 100
  end <- Sys.time()
  cat("\n N3+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "N3+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 2-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac1 = 1 - stringdist(nomination, reference, method = "jaccard", q = 1)
  end <- Sys.time()
  cat("\n JAC1 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "JAC1", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac1plus = string_compare(nomination, reference, method = "jaccard", q = 1) / 100
  end <- Sys.time()
  cat("\n JAC1+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "JAC1+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 2-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac2 = 1 - stringdist(nomination, reference, method = "jaccard", q = 2)
  end <- Sys.time()
  cat("\n JAC2 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
```

```
speed <- rbind(speed, data.frame(feature = "JAC2", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac2plus = string_compare(nomination, reference, method = "jaccard", q = 2) / 100
  end <- Sys.time()
  cat("\n JAC2+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "JAC2+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features 3-GRAM ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac3 = 1 - stringdist(nomination, reference, method = "jaccard", q = 3)
  end <- Sys.time()
  cat("\n JAC3 \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "JAC3", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$jac3plus = string_compare(nomination, reference, method = "jaccard", q = 3) / 100
  end <- Sys.time()
  cat("\n JAC3+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "JAC3+", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features INIT ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$init = initial_similarity(nomination, reference)
  end <- Sys.time()
  cat("\n INIT \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "INIT", speed = n_m * n_i / t, stringsAsFactors = F))

# Generating advanced features PHONETIC ----
t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$phon = phonetic_similarity(nomination, reference)
  end <- Sys.time()
  cat("\n PHON \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}
speed <- rbind(speed, data.frame(feature = "PHON", speed = n_m * n_i / t, stringsAsFactors = F))

t <- 0
for(i in 1:n_i){
  start <- Sys.time()
  data$phonplus = string_compare(nomination, reference, method = "soundex") / 100
  end <- Sys.time()
  cat("\n PHON+ \n")
  t <- t + as.numeric(difftime(end, start, units = "secs"))
}

```

```
318 # This data frame contains the speed of each feature in dyads / second
319 speed <- rbind(speed, data.frame(feature = "PHON+", speed = n_m * n_i / t, stringsAsFactors = F))
320
321 # Separate matches from non-matches
322 data_pos <- data[data$match == 1,]
323 data_neg <- data[data$match == 0,]
324
325 # Find the lowest value of each feature for the matches, then find what amount of non-matches fall
        below that value
326 feature_blocking <- data.frame(feature = NA, min = NA, max = NA, blocking = NA, stringsAsFactors = F
        )[0,]
327 for(i in 4:29){
328   feature <- colnames(data_pos)[i]
329   min <- min(data_pos[,i])
330   max <- max(data_pos[,i])
331   blocking <- length(data_neg$match[data_neg[,i] < min]) / nrow(data_neg)
332   add <- data.frame(feature = feature, min = min, max = max, blocking = blocking, stringsAsFactors =
          F)
333   feature_blocking <- rbind(feature_blocking, add)
334 }
335
336 save(feature_blocking, file = "./data/2021_08_22_feature_blocking_table.rda")
337
338
339 # END OF SCRIPT
```

Appendix I Colombia Data Sample

The table in this appendix (begins on the next page) contains a sample of the empirical data that we have used. We have included 898 dyads, with roughly 40% representing a match, and 60% representing a non-match. We have left out mostly non-matches, since the full set of dyads is made up of only 0.2% matches and 99.8% non-matches. In total, we have generated 309,246 dyads and their accompanying features from the empirical data set. The complete dataset is available on the github repository[22] that accompanies this thesis.

[22] https://github.com/errslima/EnzoBachelorThesis

match	jw	cos1	cos2	cos3	init	lv	lcs	n2	n3	jwplus	phon	lvplus	lcsplus	cos1plus	cos2plus	cos3plus	jac1plus	jac2plus	jac3plus
1	0.857	0.92	0.771	0.725	0.667	0.571	0.786	0.786	0.786	1	0.009	1	1	1	1	1	1	1	1
1	0.762	0.966	0.923	0.914	0.625	0.5	0.75	0.75	0.75	1	0.013	1	1	1	1	1	1	1	1
1	0.733	0.63	0.306	0.241	0.333	0.2	0.6	0.56	0.56	0.748	0.017	0.556	0.556	0.979	0.979	0.979	0.625	0.333	0.25
1	0.899	0.93	0.836	0.827	0.833	0.75	0.854	0.854	0.854	0.8	0.014	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
1	0.714	0.717	0.429	0.277	0.25	0.143	0.571	0.571	0.571	1	0.014	1	1	1	1	1	1	1	1
1	0.885	0.943	0.836	0.791	0.667	0.654	0.827	0.827	0.827	1	0.009	1	1	1	1	1	1	1	1
1	0.695	0.892	0.736	0.63	0.667	0.619	0.786	0.738	0.69	0.911	0.02	0.833	0.75	0.989	0.989	0.989	0.8	0.643	0.583
1	0.738	0.792	0.696	0.623	0.5	0.5	0.75	0.75	0.714	1	0.009	1	1	1	1	1	1	1	1
1	0.873	0.945	0.8	0.761	0.667	0.619	0.81	0.81	0.81	1	0.009	1	1	1	1	1	1	1	1
1	0.774	0.944	0.7	0.57	0.667	0.583	0.792	0.75	0.667	0.979	0.015	0.938	0.938	0.997	0.997	0.997	1	0.812	0.688
1	0.824	0.936	0.792	0.7	0.75	0.655	0.828	0.793	0.759	0.978	0.005	0.933	0.933	0.993	0.993	0.993	0.933	0.8	0.75
1	0.737	0.953	0.786	0.635	0.7	0.655	0.81	0.776	0.707	0.944	0.008	0.917	0.833	0.979	0.979	0.979	0.867	0.833	0.778
1	0.683	0.767	0.471	0.385	0.667	0.542	0.688	0.562	0.521	0.789	0.009	0.717	0.483	0.971	0.971	0.971	0.717	0.381	0.196
1	0.778	0.802	0.582	0.522	0.333	0.333	0.667	0.667	0.667	1	0.017	1	1	1	1	1	1	1	1
1	0.798	0.874	0.625	0.588	0.5	0.393	0.696	0.696	0.696	1	0.009	1	1	1	1	1	1	1	1
1	0.773	0.743	0.593	0.511	0.333	0.32	0.66	0.66	0.66	1	0.017	1	1	1	1	1	1	1	1
1	0.736	0.566	0.49	0.385	0.25	0.207	0.603	0.603	0.603	1	0.014	1	1	1	1	1	1	1	1
1	0.808	0.887	0.711	0.537	0.667	0.526	0.763	0.763	0.658	0.976	0.009	0.929	0.929	0.995	0.995	0.995	0.929	0.917	0.9
1	0.889	0.9	0.804	0.712	0.667	0.667	0.833	0.833	0.778	0.976	0.009	0.929	0.929	0.997	0.997	0.997	1	0.917	0.75
1	0.911	0.958	0.871	0.845	0.75	0.733	0.867	0.867	0.867	1	0.005	1	1	1	1	1	1	1	1
1	0.867	0.894	0.79	0.745	0.833	0.6	0.8	0.8	0.8	1	0.009	1	1	1	1	1	1	1	1
1	0.684	0.935	0.775	0.72	0.5	0.552	0.776	0.776	0.776	1	0.017	1	1	1	1	1	1	1	1
1	0.736	0.952	0.762	0.685	0.5	0.531	0.766	0.766	0.766	1	0.016	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.889	0.944	0.808	0.798	0.833	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.779	0.9	0.677	0.474	0.667	0.417	0.646	0.604	0.562	0.795	0.012	0.571	0.5	0.976	0.976	0.976	0.75	0.545	0.5
1	0.628	0.742	0.544	0.43	0.25	0.241	0.621	0.621	0.621	1	0.02	1	1	1	1	1	1	1	1
1	0.773	0.832	0.606	0.5	0.333	0.318	0.659	0.659	0.659	1	0.017	1	1	1	1	1	1	1	1
1	0.686	0.746	0.433	0.33	0.333	0.304	0.652	0.609	0.565	0.963	0.023	0.889	0.889	0.995	0.995	0.995	0.875	0.667	0.444
1	0.69	0.773	0.284	0.088	0.25	0.214	0.589	0.518	0.482	0.78	0.014	0.625	0.375	0.971	0.971	0.971	0.5	0.3	0.1
1	0.884	0.934	0.818	0.787	0.667	0.652	0.826	0.826	0.826	1	0.009	1	1	1	1	1	1	1	1
1	0.822	0.93	0.692	0.636	0.667	0.55	0.75	0.7	0.7	0.917	0.013	0.75	0.75	0.969	0.969	0.969	0.8	0.6	0.5
1	0.975	0.985	0.929	0.88	1	0.963	0.963	0.926	0.889	0.975	0	0.963	0.926	0.996	0.996	0.996	0.933	0.867	0.8
1	0.663	0.946	0.774	0.611	0.625	0.485	0.727	0.697	0.667	0.923	0.019	0.938	0.875	0.995	0.995	0.995	0.929	0.778	0.667
1	0.61	0.87	0.469	0.334	0.625	0.5	0.643	0.5	0.464	0.745	0.015	0.548	0.417	0.964	0.964	0.964	0.6	0.4	0.37
1	0.695	0.714	0.436	0.329	0.333	0.348	0.609	0.522	0.522	0.77	0.016	0.571	0.5	0.95	0.95	0.95	0.611	0.5	0.5
1	0.695	0.903	0.733	0.694	0.625	0.517	0.759	0.759	0.759	1	0.018	1	1	1	1	1	1	1	1
1	0.885	0.944	0.8	0.791	0.75	0.654	0.827	0.827	0.827	1	0.005	1	1	1	1	1	1	1	1
1	0.888	0.973	0.888	0.815	0.875	0.677	0.823	0.79	0.758	0.944	0.005	0.917	0.833	0.979	0.979	0.979	0.867	0.833	0.778
1	0.879	0.894	0.787	0.775	0.667	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.879	0.884	0.81	0.775	0.833	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.88	0.95	0.779	0.724	0.833	0.64	0.82	0.78	0.78	0.958	0.009	0.938	0.875	0.996	0.996	0.996	0.929	0.875	0.857
1	0.756	0.915	0.695	0.532	0.667	0.583	0.771	0.729	0.646	0.904	0.009	0.795	0.723	0.984	0.984	0.984	0.786	0.616	0.56
1	0.742	0.566	0.417	0.387	0.333	0.227	0.614	0.614	0.614	1	0.017	1	1	1	1	1	1	1	1
1	0.694	0.77	0.41	0.125	0.333	0.333	0.667	0.556	0.5	0.869	0.017	0.714	0.571	0.989	0.989	0.989	0.857	0.375	0.125
1	0.784	0.716	0.559	0.516	0.333	0.353	0.676	0.676	0.676	1	0.017	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.717	0.938	0.483	0.304	0.625	0.433	0.683	0.583	0.517	0.896	0.012	0.789	0.689	0.985	0.985	0.985	0.9	0.527	0.386
1	0.893	0.987	0.973	0.951	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.636	0.703	0.369	0.171	0.333	0.211	0.605	0.553	0.553	0.833	0.025	0.75	0.5	0.981	0.981	0.981	0.8	0.5	0.333
1	0.709	0.813	0.471	0.288	0.5	0.379	0.655	0.517	0.448	0.747	0.02	0.562	0.438	0.98	0.98	0.98	0.667	0.357	0.188
1	0.889	0.923	0.826	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.893	0.928	0.803	0.761	0.875	0.679	0.839	0.804	0.804	0.968	0.005	0.952	0.905	0.996	0.996	0.996	0.944	0.905	0.889
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.984	0.988	0.93	0.865	1	0.952	0.976	0.929	0.881	0.972	0	0.917	0.917	0.989	0.989	0.989	0.917	0.889	0.833
1	0.724	0.888	0.775	0.695	0.5	0.516	0.758	0.758	0.758	1	0.017	1	1	1	1	1	1	1	1
1	0.867	0.961	0.817	0.778	0.667	0.6	0.8	0.8	0.8	1	0.009	1	1	1	1	1	1	1	1
1	0.872	0.934	0.767	0.655	0.667	0.615	0.808	0.769	0.731	0.981	0.009	0.944	0.944	0.998	0.998	0.998	0.929	0.833	0.722
1	0.789	0.939	0.81	0.688	0.75	0.72	0.86	0.82	0.74	0.986	0.012	0.958	0.958	0.998	0.998	0.998	0.952	0.952	0.944
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1

1	0.901	0.945	0.857	0.825	0.75	0.704	0.852	0.852	0.852	1	0.005	1	1	1	1	1	1	1
1	0.909	0.979	0.901	0.837	0.667	0.727	0.864	0.864	0.864	1	0.009	1	1	1	1	1	1	1
1	0.873	0.945	0.8	0.761	0.667	0.619	0.81	0.81	0.81	1	0.009	1	1	1	1	1	1	1
1	0.75	0.921	0.786	0.719	0.5	0.517	0.759	0.759	0.759	1	0.014	1	1	1	1	1	1	1
1	0.87	0.935	0.793	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1
1	0.77	0.927	0.8	0.692	0.667	0.619	0.81	0.81	0.762	1	0.011	1	1	1	1	1	1	1
1	0.672	0.923	0.566	0.414	0.333	0.5	0.688	0.646	0.562	0.889	0.023	0.812	0.625	0.984	0.984	0.984	0.8	0.636	0.6
1	0.89	0.981	0.834	0.763	1	0.909	0.909	0.818	0.773	0.926	0.005	0.889	0.778	0.991	0.991	0.991	0.905	0.75	0.714
1	0.545	0.595	0.34	0.249	0.25	0.194	0.581	0.548	0.548	0.905	0.028	0.857	0.714	0.98	0.98	0.98	0.75	0.5	0.429
1	0.87	0.943	0.816	0.756	0.833	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.767	0.934	0.72	0.629	0.5	0.5	0.75	0.75	0.717	1	0.012	1	1	1	1	1	1	1	1
1	0.776	0.87	0.726	0.71	0.333	0.591	0.773	0.773	0.773	0.944	0.013	0.917	0.833	0.989	0.989	0.989	0.917	0.833	0.8
1	0.879	0.884	0.81	0.775	0.833	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.88	0.96	0.83	0.78	0.667	0.64	0.82	0.82	0.82	1	0.009	1	1	1	1	1	1	1	1
1	0.885	0.943	0.836	0.791	0.667	0.654	0.827	0.827	0.827	1	0.009	1	1	1	1	1	1	1	1
1	0.875	0.928	0.818	0.769	0.667	0.625	0.812	0.812	0.812	1	0.009	1	1	1	1	1	1	1	1
1	0.839	0.952	0.613	0.532	0.667	0.542	0.771	0.688	0.646	0.952	0.013	0.857	0.857	0.992	0.992	0.992	0.857	0.75	0.714
1	0.725	0.921	0.756	0.596	0.875	0.607	0.768	0.732	0.661	0.87	0.005	0.778	0.639	0.957	0.957	0.957	0.744	0.639	0.533
1	0.783	0.884	0.79	0.745	0.833	0.6	0.8	0.8	0.8	1	0.012	1	1	1	1	1	1	1	1
1	0.652	0.762	0.383	0.385	0.25	0.207	0.603	0.603	0.603	1	0.014	1	1	1	1	1	1	1	1
1	0.76	0.738	0.48	0.466	0.5	0.28	0.64	0.64	0.64	1	0.017	1	1	1	1	1	1	1	1
1	0.627	0.754	0.454	0.378	0.5	0.214	0.607	0.607	0.607	1	0.023	1	1	1	1	1	1	1	1
1	0.758	0.662	0.488	0.447	0.333	0.273	0.636	0.636	0.636	1	0.017	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.903	0.985	0.962	0.933	0.875	0.71	0.855	0.855	0.855	1	0.005	1	1	1	1	1	1	1	1
1	0.817	0.943	0.818	0.787	0.667	0.652	0.826	0.826	0.826	1	0.014	1	1	1	1	1	1	1	1
1	0.722	0.776	0.547	0.456	0.667	0.5	0.667	0.583	0.583	0.77	0.017	0.643	0.5	0.956	0.956	0.956	0.6	0.5	0.5
1	0.783	0.91	0.8	0.692	0.667	0.619	0.81	0.81	0.762	1	0.012	1	1	1	1	1	1	1	1
1	0.487	0.587	0	0	0	0.105	0.447	0.342	0.395	0.512	0.04	0.125	0	0.934	0.934	0.934	0.3	0	0
1	0.736	0.952	0.762	0.685	0.5	0.531	0.766	0.766	0.766	1	0.016	1	1	1	1	1	1	1	1
1	0.767	0.934	0.72	0.629	0.5	0.5	0.75	0.75	0.717	1	0.012	1	1	1	1	1	1	1	1
1	0.588	0.905	0.598	0.533	0.5	0.407	0.704	0.704	0.667	1	0.022	1	1	1	1	1	1	1	1
1	0.754	0.78	0.422	0.35	0.333	0.522	0.717	0.543	0.5	0.812	0.017	0.625	0.5	0.969	0.969	0.969	0.667	0.538	0.5
1	0.605	0.931	0.792	0.67	0.5	0.517	0.759	0.759	0.724	1	0.02	1	1	1	1	1	1	1	1
1	0.877	0.937	0.836	0.767	0.833	0.632	0.816	0.816	0.816	1	0.009	1	1	1	1	1	1	1	1
1	0.853	0.93	0.824	0.718	0.75	0.667	0.833	0.833	0.767	0.989	0.006	0.967	0.967	0.999	0.999	0.999	1	0.963	0.889
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.77	0.927	0.8	0.692	0.667	0.619	0.81	0.81	0.762	1	0.011	1	1	1	1	1	1	1	1
1	0.889	0.944	0.826	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.682	0.962	0.799	0.632	0.625	0.485	0.742	0.712	0.682	0.943	0.019	0.938	0.938	0.998	0.998	0.998	1	0.812	0.688
1	0.831	0.932	0.867	0.825	0.75	0.704	0.852	0.852	0.852	1	0.007	1	1	1	1	1	1	1	1
1	0.831	0.899	0.679	0.567	0.667	0.565	0.761	0.717	0.674	0.952	0.009	0.929	0.857	0.992	0.992	0.992	0.857	0.75	0.625
1	0.692	0.786	0.559	0.397	0.5	0.419	0.694	0.629	0.565	0.935	0.014	0.875	0.812	0.99	0.99	0.99	0.812	0.65	0.5
1	0.767	0.841	0.486	0.384	0.25	0.517	0.724	0.621	0.552	0.838	0.011	0.714	0.661	0.978	0.978	0.978	0.711	0.644	0.6
1	0.979	0.981	0.897	0.815	1	0.938	0.969	0.906	0.844	0.979	0.008	0.938	0.938	0.996	0.996	0.996	0.938	0.812	0.688
1	0.732	0.858	0.695	0.588	0.5	0.423	0.712	0.712	0.712	1	0.013	1	1	1	1	1	1	1	1
1	0.747	0.505	0.423	0.417	0.333	0.24	0.62	0.62	0.62	1	0.017	1	1	1	1	1	1	1	1
1	0.672	0.857	0.487	0.377	0.5	0.333	0.667	0.625	0.583	0.958	0.009	0.875	0.875	0.983	0.983	0.983	0.875	0.625	0.5
1	0.786	0.761	0.555	0.5	0.5	0.357	0.679	0.679	0.679	1	0.021	1	1	1	1	1	1	1	1
1	0.844	0.953	0.748	0.625	0.5	0.586	0.776	0.741	0.707	0.95	0.009	0.905	0.857	0.985	0.985	0.985	0.905	0.857	0.833
1	0.801	0.868	0.753	0.613	0.667	0.6	0.78	0.74	0.7	0.923	0.019	0.938	0.875	0.995	0.995	0.995	0.929	0.778	0.667
1	0.663	0.963	0.774	0.611	0.625	0.485	0.727	0.697	0.667	0.923	0.012	0.938	0.875	0.995	0.995	0.995	0.929	0.778	0.667
1	0.764	0.944	0.881	0.858	0.667	0.762	0.881	0.881	0.881	1	0.02	1	1	1	1	1	1	1	1
1	0.879	0.894	0.787	0.775	0.667	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.831	0.89	0.65	0.567	0.667	0.565	0.761	0.717	0.674	0.944	0.009	0.917	0.833	0.989	0.989	0.989	0.917	0.833	0.8
1	0.711	0.652	0.371	0.255	0.333	0.24	0.62	0.58	0.54	0.887	0.023	0.875	0.875	0.995	0.995	0.995	1	0.625	0.375
1	0.797	0.927	0.738	0.642	0.667	0.609	0.804	0.761	0.717	0.979	0.015	0.938	0.938	0.997	0.997	0.997	0.929	0.812	0.688
1	0.98	0.987	0.968	0.897	1	0.941	0.971	0.971	0.912	0.981	0	0.944	0.944	0.998	0.998	0.998	1	0.933	0.8
1	0.707	0.814	0.63	0.59	0.625	0.4	0.7	0.7	0.7	1	0.012	1	1	1	1	1	1	1	1

1	0.979	0.975	0.897	0.815	1	0.938	0.969	0.906	0.844	0.976	0	0.929	0.929	0.995	0.995	0.995	0.929	0.917	0.9
1	0.737	0.82	0.666	0.581	0.667	0.591	0.773	0.727	0.682	0.933	0.014	0.9	0.8	0.985	0.985	0.985	0.9	0.8	0.75
1	0.602	0.809	0.423	0.4	0.25	0.222	0.611	0.611	0.611	1	0.023	1	1	1	1	1	1	1	1
1	0.842	0.904	0.742	0.686	0.667	0.526	0.763	0.763	0.763	1	0.009	1	1	1	1	1	1	1	1
1	0.72	0.471	0.333	0.322	0.25	0.161	0.581	0.581	0.581	1	0.014	1	1	1	1	1	1	1	1
1	0.817	0.924	0.833	0.726	0.667	0.652	0.826	0.826	0.783	1	0.012	1	1	1	1	1	1	1	1
1	0.652	0.762	0.383	0.385	0.25	0.207	0.603	0.603	0.603	1	0.014	1	1	1	1	1	1	1	1
1	0.684	0.857	0.733	0.721	0.625	0.517	0.759	0.759	0.759	1	0.018	1	1	1	1	1	1	1	1
1	0.889	0.944	0.808	0.798	0.833	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.628	0.742	0.544	0.43	0.25	0.241	0.621	0.621	0.621	1	0.02	1	1	1	1	1	1	1	1
1	0.627	0.843	0.474	0.433	0.25	0.323	0.645	0.613	0.613	0.756	0.021	0.7	0.6	0.968	0.968	0.968	0.6	0.5	0.5
1	0.909	0.979	0.901	0.837	0.667	0.727	0.864	0.864	0.864	1	0.009	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.815	0.961	0.913	0.891	0.75	0.793	0.897	0.897	0.897	1	0.011	1	1	1	1	1	1	1	1
1	0.782	0.906	0.661	0.54	0.5	0.346	0.673	0.673	0.673	1	0.017	1	1	1	1	1	1	1	1
1	0.87	0.895	0.781	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.847	0.978	0.887	0.797	0.833	0.609	0.783	0.739	0.696	0.972	0.009	0.917	0.917	0.993	0.993	0.993	0.917	0.9	0.875
1	0.764	0.944	0.881	0.858	0.667	0.762	0.881	0.881	0.881	1	0.02	1	1	1	1	1	1	1	1
1	0.651	0.945	0.779	0.694	0.625	0.517	0.759	0.759	0.759	1	0.021	1	1	1	1	1	1	1	1
1	0.952	0.963	0.884	0.811	1	0.905	0.929	0.881	0.833	0.95	0.009	0.905	0.857	0.992	0.992	0.992	0.857	0.792	0.708
1	0.732	0.834	0.728	0.627	0.667	0.625	0.792	0.75	0.708	0.911	0.009	0.917	0.833	0.989	0.989	0.989	0.917	0.714	0.571
1	0.87	0.919	0.793	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.655	0.965	0.92	0.864	0.625	0.464	0.732	0.732	0.732	1	0.018	1	1	1	1	1	1	1	1
1	0.626	0.829	0.616	0.49	0.5	0.433	0.683	0.65	0.617	0.889	0.021	0.833	0.667	0.977	0.977	0.977	0.786	0.714	0.667
1	0.903	0.983	0.969	0.946	0.875	0.71	0.855	0.855	0.855	1	0.005	1	1	1	1	1	1	1	1
1	0.806	0.9	0.623	0.567	0.5	0.478	0.717	0.674	0.674	0.911	0.013	0.75	0.667	0.99	0.99	0.99	0.833	0.625	0.571
1	0.88	0.947	0.825	0.78	0.833	0.64	0.82	0.82	0.82	1	0.009	1	1	1	1	1	1	1	1
1	0.873	0.877	0.8	0.761	0.833	0.619	0.81	0.81	0.81	1	0.009	1	1	1	1	1	1	1	1
1	0.74	0.899	0.63	0.539	0.833	0.545	0.75	0.705	0.659	0.933	0.011	0.9	0.8	0.98	0.98	0.98	0.833	0.8	0.75
1	0.875	0.928	0.818	0.769	0.667	0.625	0.812	0.812	0.812	1	0.009	1	1	1	1	1	1	1	1
1	0.732	0.834	0.728	0.627	0.667	0.625	0.792	0.75	0.708	0.911	0.009	0.917	0.833	0.989	0.989	0.989	0.917	0.714	0.571
1	0.793	0.889	0.705	0.577	0.5	0.379	0.69	0.69	0.69	1	0.009	1	1	1	1	1	1	1	1
1	0.873	0.945	0.8	0.761	0.667	0.619	0.81	0.81	0.81	1	0.009	1	1	1	1	1	1	1	1
1	0.889	0.923	0.826	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.817	0.924	0.833	0.726	0.667	0.652	0.826	0.826	0.783	1	0.012	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.88	0.898	0.841	0.78	0.667	0.64	0.82	0.82	0.82	1	0.009	1	1	1	1	1	1	1	1
1	0.889	0.944	0.808	0.798	0.833	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.87	0.935	0.793	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.831	0.9	0.736	0.693	0.5	0.565	0.761	0.761	0.761	0.944	0.013	0.917	0.833	0.99	0.99	0.99	0.833	0.833	0.8
1	0.839	0.926	0.669	0.591	0.667	0.583	0.771	0.729	0.688	0.952	0.013	0.929	0.857	0.992	0.992	0.992	0.857	0.857	0.833
1	0.903	0.985	0.962	0.933	0.875	0.71	0.855	0.855	0.855	1	0.005	1	1	1	1	1	1	1	1
1	0.925	0.975	0.894	0.83	1	0.905	0.929	0.881	0.833	0.941	0	0.889	0.833	0.993	0.993	0.993	0.867	0.762	0.722
1	0.875	0.928	0.818	0.769	0.667	0.625	0.812	0.812	0.812	1	0.009	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.732	0.885	0.491	0.363	0.667	0.522	0.696	0.565	0.522	0.792	0.011	0.625	0.5	0.969	0.969	0.969	0.636	0.538	0.5
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.787	0.827	0.577	0.552	0.333	0.36	0.68	0.68	0.68	1	0.017	1	1	1	1	1	1	1	1
1	0.763	0.923	0.797	0.736	0.625	0.538	0.769	0.769	0.769	1	0.012	1	1	1	1	1	1	1	1
1	0.773	0.758	0.49	0.5	0.333	0.318	0.659	0.659	0.659	1	0.017	1	1	1	1	1	1	1	1
1	0.847	0.945	0.615	0.484	0.667	0.524	0.762	0.667	0.619	0.967	0.009	0.8	0.8	1	1	1	1	0.571	0.5
1	0.684	0.857	0.733	0.721	0.625	0.517	0.759	0.759	0.759	1	0.018	1	1	1	1	1	1	1	1
1	0.88	0.947	0.825	0.78	0.833	0.64	0.82	0.82	0.82	1	0.009	1	1	1	1	1	1	1	1
1	0.77	0.927	0.8	0.692	0.667	0.619	0.81	0.81	0.762	1	0.011	1	1	1	1	1	1	1	1
1	0.795	0.96	0.7	0.627	0.667	0.583	0.792	0.75	0.708	0.979	0.013	0.938	0.938	0.997	0.997	0.997	0.929	0.929	0.917
1	0.893	0.987	0.973	0.951	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.826	0.934	0.808	0.741	0.667	0.667	0.833	0.833	0.792	1	0.014	1	1	1	1	1	1	1	1
1	0.979	0.975	0.897	0.815	1	0.938	0.969	0.906	0.844	0.976	0	0.929	0.929	0.995	0.995	0.995	0.929	0.917	0.9
1	0.703	0.854	0.721	0.648	0.5	0.538	0.769	0.769	0.731	1	0.02	1	1	1	1	1	1	1	1
1	0.78	0.809	0.707	0.602	0.833	0.565	0.761	0.717	0.674	0.911	0.015	0.833	0.75	0.981	0.981	0.981	0.833	0.643	0.583
1	0.879	0.884	0.81	0.775	0.833	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.768	0.906	0.588	0.42	0.833	0.632	0.816	0.658	0.553	0.925	0.014	0.857	0.786	0.986	0.986	0.986	0.857	0.611	0.562

1	0.893	0.987	0.973	0.951	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.801	0.87	0.545	0.387	0.667	0.545	0.727	0.636	0.545	0.889	0.02	0.833	0.667	0.977	0.977	0.977	0.786	0.625	0.5
1	0.5	0.566	0.229	0	0	0.1	0.525	0.525	0.475	0.783	0.032	0.6	0.4	0.934	0.934	0.934	0.5	0.167	0
1	0.893	0.987	0.973	0.951	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.857	0.915	0.771	0.725	0.667	0.571	0.786	0.786	0.786	1	0.009	1	1	1	1	1	1	1	1
1	0.744	0.934	0.791	0.741	0.625	0.5	0.75	0.75	0.75	1	0.012	1	1	1	1	1	1	1	1
1	0.778	0.802	0.582	0.522	0.333	0.333	0.667	0.667	0.667	1	0.017	1	1	1	1	1	1	1	1
1	0.979	0.975	0.897	0.815	1	0.938	0.969	0.906	0.844	0.976	0	0.929	0.929	0.995	0.995	0.995	0.929	0.917	0.9
1	0.785	0.901	0.78	0.682	0.667	0.591	0.795	0.795	0.75	0.976	0.014	0.929	0.929	0.997	0.997	0.997	1	0.917	0.75
1	0.885	0.935	0.836	0.791	0.833	0.654	0.827	0.827	0.827	1	0.009	1	1	1	1	1	1	1	1
1	0.875	0.928	0.818	0.769	0.667	0.625	0.812	0.812	0.812	1	0.009	1	1	1	1	1	1	1	1
1	0.684	0.889	0.683	0.567	0.5	0.484	0.742	0.742	0.677	0.944	0.016	0.833	0.833	0.985	0.985	0.985	0.833	0.8	0.75
1	0.754	0.843	0.651	0.589	0.833	0.65	0.75	0.65	0.65	0.667	0.011	0.667	0.667	0.964	0.964	0.964	0.704	0.667	0.667
1	0.716	0.911	0.709	0.605	0.667	0.609	0.783	0.739	0.696	0.944	0.022	0.917	0.833	0.99	0.99	0.99	0.833	0.714	0.571
1	0.87	0.935	0.793	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.781	0.939	0.689	0.613	0.625	0.621	0.776	0.707	0.672	0.913	0.01	0.869	0.738	0.975	0.975	0.975	0.756	0.738	0.667
1	0.742	0.732	0.529	0.448	0.5	0.387	0.677	0.645	0.613	0.952	0.009	0.929	0.857	0.993	0.993	0.993	0.833	0.75	0.714
1	0.608	0.673	0.327	0.209	0.333	0.2	0.58	0.54	0.54	0.849	0.025	0.714	0.571	0.967	0.967	0.967	0.625	0.375	0.286
1	0.852	0.951	0.735	0.618	0.75	0.655	0.828	0.793	0.69	0.963	0.006	0.889	0.889	0.99	0.99	0.99	0.889	0.867	0.833
1	0.59	0.879	0.742	0.709	0.625	0.5	0.75	0.75	0.75	1	0.021	1	1	1	1	1	1	1	1
1	0.965	0.966	0.889	0.824	1	0.947	0.947	0.895	0.842	0.944	0	0.917	0.833	0.979	0.979	0.979	0.867	0.833	0.778
1	0.59	0.879	0.742	0.709	0.625	0.5	0.75	0.75	0.75	1	0.021	1	1	1	1	1	1	1	1
1	0.778	0.68	0.488	0.5	0.333	0.333	0.667	0.667	0.667	1	0.017	1	1	1	1	1	1	1	1
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1	1
1	0.722	0.924	0.62	0.456	0.667	0.625	0.792	0.667	0.583	0.935	0.019	0.85	0.8	0.996	0.996	0.996	0.889	0.615	0.5
1	0.893	0.95	0.861	0.808	0.667	0.68	0.84	0.84	0.84	1	0.009	1	1	1	1	1	1	1	1
1	0.763	0.938	0.693	0.646	0.625	0.414	0.707	0.707	0.707	1	0.011	1	1	1	1	1	1	1	1
1	0.712	0.808	0.624	0.515	0.5	0.452	0.71	0.677	0.645	0.952	0.014	0.929	0.857	0.992	0.992	0.992	0.929	0.75	0.625
1	0.877	0.937	0.836	0.767	0.833	0.632	0.816	0.816	0.816	1	0.009	1	1	1	1	1	1	1	1
1	0.674	0.915	0.753	0.695	0.375	0.517	0.741	0.741	0.741	0.952	0.021	0.929	0.857	0.992	0.992	0.992	0.929	0.857	0.833
1	0.84	0.865	0.618	0.581	0.75	0.621	0.776	0.638	0.638	0.869	0.009	0.75	0.667	0.988	0.988	0.988	0.778	0.667	0.667
1	0.897	0.963	0.849	0.816	0.75	0.69	0.845	0.845	0.845	1	0.005	1	1	1	1	1	1	1	1
1	0.7	0.879	0.664	0.385	0.667	0.375	0.625	0.583	0.458	0.726	0.012	0.509	0.438	0.976	0.976	0.976	0.75	0.345	0.167
1	0.59	0.891	0.713	0.58	0.5	0.448	0.724	0.724	0.69	1	0.019	1	1	1	1	1	1	1	1
1	0.85	0.943	0.808	0.707	0.667	0.55	0.775	0.775	0.775	1	0.009	1	1	1	1	1	1	1	1
1	0.663	0.946	0.774	0.611	0.625	0.485	0.727	0.697	0.667	0.923	0.019	0.938	0.875	0.995	0.995	0.995	0.929	0.778	0.667
1	0.896	0.965	0.871	0.802	0.833	0.688	0.844	0.844	0.844	1	0.009	1	1	1	1	1	1	1	1
1	0.985	0.989	0.934	0.872	1	0.955	0.977	0.932	0.886	0.983	0	0.95	0.95	0.997	0.997	0.997	1	0.85	0.812
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1	1
1	0.925	0.975	0.894	0.83	1	0.905	0.929	0.881	0.833	0.941	0	0.889	0.833	0.993	0.993	0.993	0.867	0.762	0.722
1	0.732	0.933	0.842	0.7	0.667	0.609	0.804	0.848	0.761	0.976	0.019	0.929	0.929	0.996	0.996	0.996	0.917	0.917	0.9
1	0.875	0.943	0.823	0.769	0.667	0.625	0.812	0.812	0.812	1	0.009	1	1	1	1	1	1	1	1
1	0.889	0.917	0.84	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.893	0.987	0.973	0.951	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.854	0.907	0.671	0.627	0.667	0.625	0.792	0.708	0.708	0.926	0.009	0.889	0.778	0.99	0.99	0.99	0.889	0.727	0.7
1	0.884	0.934	0.818	0.787	0.667	0.652	0.826	0.826	0.826	1	0.009	1	1	1	1	1	1	1	1
1	0.885	0.943	0.836	0.791	0.667	0.654	0.827	0.827	0.827	1	0.009	1	1	1	1	1	1	1	1
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1	1
1	0.925	0.975	0.894	0.83	1	0.905	0.929	0.881	0.833	0.941	0	0.889	0.833	0.993	0.993	0.993	0.867	0.762	0.722
1	0.797	0.927	0.738	0.642	0.667	0.609	0.804	0.761	0.717	0.979	0.015	0.938	0.938	0.997	0.997	0.997	0.929	0.812	0.688
1	0.754	0.804	0.546	0.434	0.667	0.455	0.659	0.568	0.568	0.756	0.013	0.571	0.5	0.967	0.967	0.967	0.636	0.5	0.5
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.879	0.884	0.81	0.775	0.833	0.636	0.818	0.818	0.818	1	0.009	1	1	1	1	1	1	1	1
1	0.831	0.9	0.736	0.693	0.5	0.565	0.761	0.761	0.761	0.944	0.013	0.917	0.833	0.99	0.99	0.99	0.833	0.833	0.8
1	0.893	0.951	0.816	0.809	0.75	0.679	0.839	0.839	0.839	0.75	0.005	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
1	0.667	0.815	0.586	0.481	0.333	0.481	0.722	0.648	0.611	0.878	0.027	0.722	0.667	0.984	0.984	0.984	0.778	0.583	0.545
1	0.847	0.945	0.554	0.553	0.667	0.571	0.762	0.667	0.667	0.976	0.009	0.857	0.857	1	1	1	1	0.667	0.625
1	0.725	0.789	0.588	0.473	0.333	0.417	0.646	0.646	0.604	0.728	0.022	0.5	0.5	0.939	0.939	0.939	0.562	0.5	0.5
1	0.68	0.883	0.653	0.519	0.7	0.405	0.649	0.622	0.568	0.562	0.015	0.543	0.486	0.761	0.761	0.761	0.525	0.44	0.4
1	0.893	0.962	0.843	0.809	0.875	0.679	0.839	0.839	0.839	1	0.005	1	1	1	1	1	1	1	1
1	0.736	0.787	0.435	0.371	0.25	0.207	0.603	0.603	0.603	1	0.014	1	1	1	1	1	1	1	1
1	0.368	0.519	0.082	0	0	0.103	0.5	0.466	0.466	0.556	0.033	0.167	0	0.935	0.935	0.935	0.333	0.091	0

1	0.558	0.554	0	0	0	0.125	0.547	0.453	0.484	0.55	0.025	0.125	0	0.946	0.946	0.946	0.333	0	0
1	0.765	0.72	0.522	0.483	0.5	0.625	0.719	0.594	0.594	0.692	0.033	0.625	0.5	0.963	0.963	0.963	0.625	0.5	0.5
1	0.87	0.895	0.781	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.716	0.946	0.7	0.491	0.667	0.577	0.769	0.731	0.615	0.873	0.016	0.786	0.643	0.988	0.988	0.988	0.786	0.611	0.5
1	0.48	0.525	0	0	0	0.148	0.537	0.426	0.463	0.54	0.04	0.286	0	0.962	0.962	0.962	0.375	0	0
1	0.889	0.951	0.856	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.706	0.694	0.333	0.293	0.333	0.261	0.609	0.565	0.565	0.81	0.017	0.714	0.429	0.968	0.968	0.968	0.714	0.5	0.429
1	0.733	0.765	0.392	0.361	0.25	0.2	0.6	0.6	0.6	1	0.014	1	1	1	1	1	1	1	1
1	0.786	0.806	0.555	0.5	0.5	0.357	0.679	0.679	0.679	1	0.021	1	1	1	1	1	1	1	1
1	0.725	0.596	0.369	0.309	0.333	0.174	0.587	0.587	0.587	0.889	0.017	0.667	0.667	0.969	0.969	0.969	0.667	0.6	0.5
1	0.704	0.952	0.897	0.88	0.625	0.419	0.71	0.71	0.71	1	0.012	1	1	1	1	1	1	1	1
1	0.794	0.916	0.828	0.728	0.667	0.667	0.833	0.833	0.786	1	0.014	1	1	1	1	1	1	1	1
1	0.66	0.556	0.093	0	0.25	0.143	0.554	0.482	0.482	0.878	0.014	0.5	0.5	0.997	0.997	0.997	1	0.125	0
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.795	0.96	0.7	0.627	0.667	0.583	0.792	0.75	0.708	0.979	0.013	0.938	0.938	0.997	0.997	0.997	0.929	0.929	0.917
1	0.873	0.927	0.775	0.761	0.667	0.619	0.81	0.81	0.81	1	0.009	1	1	1	1	1	1	1	1
1	0.704	0.952	0.897	0.88	0.625	0.419	0.71	0.71	0.71	1	0.012	1	1	1	1	1	1	1	1
1	0.667	0.895	0.672	0.577	0.5	0.5	0.75	0.717	0.683	0.948	0.018	0.929	0.929	0.997	0.997	0.997	1	0.786	0.643
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.877	0.937	0.836	0.767	0.833	0.632	0.816	0.816	0.816	1	0.009	1	1	1	1	1	1	1	1
1	0.76	0.816	0.596	0.456	0.667	0.458	0.667	0.625	0.583	0.795	0.016	0.643	0.5	0.976	0.976	0.976	0.75	0.545	0.5
1	0.903	0.975	0.905	0.854	0.75	0.71	0.855	0.855	0.855	1	0.005	1	1	1	1	1	1	1	1
1	0.9	0.966	0.863	0.824	0.875	0.7	0.85	0.85	0.85	1	0.005	1	1	1	1	1	1	1	1
1	0.789	0.959	0.845	0.795	0.75	0.724	0.862	0.862	0.828	1	0.01	1	1	1	1	1	1	1	1
1	0.973	0.993	0.979	0.934	1	0.96	0.98	0.98	0.94	0.988	0	0.963	0.963	0.999	0.999	0.999	1	0.958	0.952
1	0.713	0.9	0.735	0.546	0.75	0.586	0.759	0.724	0.621	0.909	0.005	0.841	0.738	0.985	0.985	0.985	0.833	0.595	0.476
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.824	0.848	0.757	0.688	0.667	0.6	0.78	0.74	0.74	0.958	0.009	0.938	0.875	0.994	0.994	0.994	1	0.778	0.75
1	0.684	0.857	0.733	0.721	0.625	0.517	0.759	0.759	0.759	1	0.018	1	1	1	1	1	1	1	1
1	0.87	0.935	0.793	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.939	0.946	0.822	0.667	1	0.909	0.909	0.818	0.727	0.944	0	0.917	0.833	0.99	0.99	0.99	0.833	0.714	0.571
1	0.889	0.923	0.826	0.798	0.667	0.667	0.833	0.833	0.833	1	0.009	1	1	1	1	1	1	1	1
1	0.703	0.973	0.907	0.877	0.75	0.516	0.758	0.758	0.758	1	0.016	1	1	1	1	1	1	1	1
1	0.732	0.858	0.695	0.588	0.5	0.423	0.712	0.712	0.712	1	0.013	1	1	1	1	1	1	1	1
1	0.798	0.874	0.625	0.588	0.5	0.393	0.696	0.696	0.696	1	0.009	1	1	1	1	1	1	1	1
1	0.672	0.785	0.535	0.412	0.5	0.419	0.677	0.613	0.581	0.905	0.014	0.857	0.714	0.983	0.983	0.983	0.812	0.6	0.5
1	0.791	0.918	0.754	0.682	0.75	0.621	0.793	0.759	0.724	0.907	0.005	0.833	0.75	0.965	0.965	0.965	0.8	0.75	0.667
1	0.639	0.836	0.333	0.096	0.5	0.323	0.613	0.484	0.355	0.738	0.016	0.562	0.312	0.975	0.975	0.975	0.661	0.192	0
1	0.77	0.909	0.697	0.591	0.5	0.464	0.732	0.732	0.696	0.972	0.013	0.917	0.917	0.998	0.998	0.998	1	0.9	0.875
1	0.684	0.857	0.733	0.721	0.625	0.517	0.759	0.759	0.759	1	0.018	1	1	1	1	1	1	1	1
1	0.87	0.895	0.781	0.756	0.667	0.609	0.804	0.804	0.804	1	0.009	1	1	1	1	1	1	1	1
1	0.901	0.974	0.87	0.825	0.75	0.704	0.852	0.852	0.852	1	0.005	1	1	1	1	1	1	1	1
1	0.778	0.831	0.4	0.348	0.5	0.571	0.679	0.464	0.464	0.756	0.025	0.625	0.5	0.967	0.967	0.967	0.636	0.5	0.5
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0.813	0.957	0.694	0.615	0.667	0.583	0.792	0.75	0.708	0.976	0.012	0.929	0.929	0.996	0.996	0.996	0.917	0.786	0.75
1	0.885	0.943	0.836	0.791	0.667	0.654	0.827	0.827	0.827	1	0.009	1	1	1	1	1	1	1	1
1	0.831	0.9	0.736	0.693	0.5	0.565	0.761	0.761	0.761	0.944	0.013	0.917	0.833	0.99	0.99	0.99	0.833	0.833	0.8
1	0.765	0.909	0.7	0.627	0.667	0.625	0.792	0.75	0.708	0.952	0.02	0.929	0.857	0.992	0.992	0.992	0.857	0.857	0.833
1	0.896	0.926	0.871	0.833	0.833	0.688	0.844	0.844	0.844	1	0.009	1	1	1	1	1	1	1	1
0	0.584	0.663	0.167	0.093	0	0.276	0.517	0.379	0.345	0.619	0.029	0.313	0.111	0.966	0.966	0.966	0.455	0.125	0.091
0	0.612	0.791	0.598	0.482	0.333	0.48	0.7	0.66	0.62	0.781	0.027	0.643	0.5	0.969	0.969	0.969	0.688	0.556	0.5
0	0.524	0.639	0.16	0.059	0.25	0.25	0.458	0.375	0.333	0.554	0.02	0.288	0	0.933	0.933	0.933	0.389	0.05	0
0	0.616	0.818	0.487	0.384	0.25	0.419	0.645	0.548	0.548	0.825	0.021	0.688	0.5	0.983	0.983	0.983	0.722	0.5	0.5
0	0.585	0.762	0.417	0.374	0.375	0.345	0.621	0.552	0.552	0.77	0.022	0.571	0.5	0.958	0.958	0.958	0.667	0.5	0.5
0	0.654	0.88	0.476	0.286	0.375	0.222	0.537	0.426	0.352	0.53	0.018	0.424	0.364	0.943	0.943	0.943	0.5	0.359	0.333
0	0.616	0.755	0.115	0	0	0.167	0.417	0.194	0.139	0.542	0.042	0.188	0.125	0.942	0.942	0.942	0.42	0.094	0
0	0.576	0.787	0.207	0.04	0.25	0.25	0.484	0.328	0.234	0.567	0.021	0.262	0	0.938	0.938	0.938	0.356	0.061	0
0	0.482	0.555	0.06	0	0	0.208	0.5	0.333	0.333	0.553	0.036	0.208	0	0.911	0.911	0.911	0.254	0.056	0
0	0.566	0.757	0.242	0.087	0	0.208	0.333	0.292	0.167	0.423	0.025	0.222	0.074	0.923	0.923	0.923	0.3	0.124	0.074
0	0.472	0.6	0.067	0	0.333	0.143	0.452	0.31	0.31	0.492	0.026	0.146	0	0.918	0.918	0.918	0.325	0	0
0	0.546	0.821	0.062	0	0	0.217	0.5	0.326	0.326	0.467	0.038	0.214	0	0.946	0.946	0.946	0.397	0.056	0

0	0.584	0.599	0.099	0	0.333	0.192	0.442	0.327	0.288	0.539	0.038	0.161	0	0.955	0.955	0.955	0.258	0.05	0
0	0.594	0.648	0.128	0	0.5	0.208	0.438	0.312	0.271	0.526	0.022	0.205	0	0.934	0.934	0.934	0.386	0.05	0
0	0.546	0.592	0.333	0.162	0	0.28	0.48	0.4	0.36	0.627	0.033	0.188	0.111	0.943	0.943	0.943	0.433	0.182	0.15
0	0.505	0.657	0	0	0	0.172	0.483	0.31	0.345	0.448	0.032	0.143	0	0.912	0.912	0.912	0.31	0	0
0	0.448	0.757	0.11	0	0.333	0.185	0.5	0.389	0.352	0.561	0.027	0.234	0.125	0.937	0.937	0.937	0.303	0.056	0
0	0.536	0.517	0.105	0	0.5	0.174	0.457	0.326	0.283	0.527	0.031	0.25	0.062	0.927	0.927	0.927	0.336	0.042	0
0	0.554	0.752	0.337	0.232	0.5	0.259	0.556	0.444	0.444	0.667	0.028	0.33	0.083	0.95	0.95	0.95	0.5	0.112	0
0	0.635	0.885	0.367	0.066	0.5	0.261	0.565	0.522	0.348	0.725	0.022	0.325	0.2	0.957	0.957	0.957	0.5	0.283	0.125
0	0.459	0.62	0.082	0	0.25	0.152	0.47	0.379	0.348	0.52	0.032	0.196	0.071	0.952	0.952	0.952	0.388	0.042	0
0	0.433	0.557	0.061	0	0	0.16	0.46	0.38	0.38	0.531	0.026	0.188	0	0.924	0.924	0.924	0.238	0.071	0
0	0.529	0.405	0.085	0	0.5	0.2	0.333	0.267	0.267	0.455	0.041	0.156	0	0.893	0.893	0.893	0.117	0	0
0	0.67	0.797	0.419	0.404	0.333	0.37	0.611	0.574	0.574	0.567	0.021	0.556	0.5	0.961	0.961	0.961	0.583	0.5	0.5
0	0.629	0.782	0.066	0	0	0.273	0.591	0.318	0.318	0.674	0.027	0.243	0.071	0.935	0.935	0.935	0.5	0.045	0
0	0.537	0.664	0.048	0	0.333	0.154	0.423	0.269	0.269	0.512	0.024	0.125	0	0.933	0.933	0.933	0.277	0	0
0	0.533	0.585	0.2	0.089	0	0.185	0.556	0.519	0.481	0.503	0.032	0.375	0	0.94	0.94	0.94	0.429	0.2	0.1
0	0.638	0.704	0.212	0.06	0.5	0.318	0.545	0.364	0.273	0.711	0.021	0.411	0.214	0.962	0.962	0.962	0.553	0.163	0.071
0	0.525	0.757	0.253	0.109	0.25	0.231	0.462	0.423	0.346	0.562	0.03	0.259	0.188	0.953	0.953	0.953	0.457	0.143	0.083
0	0.491	0.533	0	0	0	0.227	0.5	0.273	0.318	0.49	0.046	0.071	0	0.901	0.901	0.901	0.222	0	0
0	0.466	0.609	0.131	0	0	0.192	0.519	0.404	0.404	0.565	0.04	0.111	0	0.947	0.947	0.947	0.444	0.071	0
0	0.578	0.748	0.179	0	0	0.28	0.56	0.44	0.36	0.642	0.031	0.33	0.062	0.959	0.959	0.959	0.536	0.106	0
0	0.55	0.676	0.14	0	0	0.308	0.558	0.365	0.288	0.679	0.026	0.286	0.071	0.944	0.944	0.944	0.417	0.095	0
0	0.573	0.672	0.123	0.069	0.333	0.125	0.344	0.219	0.219	0.42	0.031	0.222	0.167	0.885	0.885	0.885	0.188	0.1	0.056
0	0.573	0.84	0.169	0.049	0	0.25	0.518	0.375	0.304	0.571	0.032	0.254	0	0.955	0.955	0.955	0.5	0.122	0.042
0	0.504	0.693	0.065	0	0	0.261	0.522	0.348	0.348	0.57	0.039	0.292	0	0.935	0.935	0.935	0.299	0.062	0
0	0.502	0.56	0.055	0	0	0.19	0.452	0.214	0.214	0.565	0.033	0.125	0	0.931	0.931	0.931	0.262	0.045	0
0	0.527	0.592	0.196	0.082	0	0.167	0.55	0.483	0.483	0.619	0.019	0.5	0.125	0.95	0.95	0.95	0.4	0.182	0.1
0	0.517	0.489	0.102	0	0.375	0.233	0.533	0.367	0.367	0.491	0.019	0.214	0	0.919	0.919	0.919	0.325	0	0
0	0.572	0.678	0.22	0	0.5	0.259	0.481	0.296	0.185	0.497	0.029	0.194	0.042	0.932	0.932	0.932	0.226	0.056	0
0	0.501	0.416	0	0	0	0.136	0.523	0.386	0.432	0.524	0.04	0.286	0	0.921	0.921	0.921	0.222	0	0
0	0.533	0.594	0.048	0	0.25	0.231	0.462	0.308	0.308	0.591	0.021	0.238	0	0.924	0.924	0.924	0.347	0.05	0
0	0.47	0.519	0	0	0	0.143	0.524	0.429	0.476	0.565	0.032	0.143	0	0.939	0.939	0.939	0.375	0	0
0	0.53	0.81	0.137	0	0	0.2	0.42	0.3	0.26	0.539	0.026	0.199	0	0.947	0.947	0.947	0.237	0.045	0
0	0.527	0.599	0.106	0.06	0	0.273	0.5	0.273	0.273	0.571	0.026	0.278	0.056	0.927	0.927	0.927	0.333	0.056	0
0	0.645	0.829	0.22	0	0.375	0.233	0.55	0.383	0.317	0.578	0.024	0.286	0.071	0.959	0.959	0.959	0.512	0.056	0
0	0.573	0.841	0.342	0.057	0	0.35	0.525	0.325	0.175	0.617	0.022	0.267	0.074	0.907	0.907	0.907	0.313	0.088	0
0	0.509	0.759	0.123	0	0.25	0.174	0.478	0.348	0.304	0.625	0.028	0.312	0.125	0.927	0.927	0.927	0.381	0.056	0
0	0.562	0.684	0.039	0	0	0.226	0.532	0.339	0.339	0.63	0.017	0.311	0.056	0.948	0.948	0.948	0.5	0.045	0
0	0.454	0.379	0.103	0	0.333	0.167	0.375	0.333	0.292	0.48	0.031	0.205	0	0.897	0.897	0.897	0.211	0.042	0
0	0.49	0.403	0.075	0	0	0.185	0.556	0.444	0.444	0.54	0.025	0.286	0	0.936	0.936	0.936	0.286	0.1	0
0	0.454	0.369	0.065	0	0.333	0.158	0.368	0.263	0.263	0.498	0.026	0.062	0	0.901	0.901	0.901	0.208	0	0
0	0.614	0.777	0.06	0	0	0.286	0.476	0.238	0.238	0.648	0.035	0.214	0	0.944	0.944	0.944	0.465	0.045	0
0	0.463	0.5	0	0	0	0.174	0.435	0.304	0.348	0.454	0.035	0.196	0	0.923	0.923	0.923	0.27	0	0
0	0.487	0.615	0.132	0.071	0.667	0.25	0.521	0.396	0.396	0.61	0.029	0.33	0.062	0.925	0.925	0.925	0.275	0.05	0
0	0.5	0.243	0	0	0	0.1	0.475	0.425	0.475	0.589	0.032	0.333	0	0.925	0.925	0.925	0.375	0	0
0	0.527	0.315	0	0	0	0.04	0.5	0.46	0.5	0.583	0.032	0.143	0	0.915	0.915	0.915	0.222	0	0
0	0.583	0.781	0.092	0	0	0.207	0.483	0.345	0.31	0.571	0.031	0.199	0.091	0.958	0.958	0.958	0.4	0.036	0
0	0.595	0.723	0.182	0.111	0.833	0.238	0.429	0.238	0.238	0.519	0.021	0.21	0.048	0.912	0.912	0.912	0.269	0.037	0
0	0.555	0.746	0.275	0	0	0.172	0.483	0.483	0.345	0.575	0.039	0.182	0.136	0.93	0.93	0.93	0.299	0.125	0
0	0.626	0.764	0.451	0.333	0.833	0.25	0.5	0.45	0.4	0.578	0.022	0.429	0.333	0.919	0.919	0.919	0.458	0.333	0.333
0	0.545	0.614	0	0	0	0.182	0.545	0.409	0.455	0.625	0.032	0.25	0	0.944	0.944	0.944	0.333	0	0
0	0.557	0.767	0.101	0	0	0.24	0.46	0.34	0.3	0.531	0.03	0.205	0	0.934	0.934	0.934	0.314	0.05	0
0	0.498	0.574	0	0	0	0.107	0.518	0.411	0.446	0.54	0.03	0.25	0	0.952	0.952	0.952	0.333	0	0
0	0.636	0.732	0	0	0.333	0.1	0.25	0.05	0.1	0.373	0.026	0.141	0	0.927	0.927	0.927	0.347	0	0
0	0.524	0.356	0.183	0	0	0.095	0.476	0.476	0.429	0.601	0.044	0.125	0	0.935	0.935	0.935	0.2	0.091	0
0	0.474	0.717	0.102	0	0	0.172	0.483	0.345	0.345	0.543	0.028	0.205	0	0.93	0.93	0.93	0.367	0.042	0
0	0.513	0.681	0.115	0	0	0.167	0.361	0.25	0.194	0.408	0.036	0.188	0.062	0.902	0.902	0.902	0.222	0.111	0
0	0.528	0.606	0.051	0	0	0.2	0.517	0.383	0.383	0.514	0.022	0.25	0	0.929	0.929	0.929	0.271	0.05	0
0	0.505	0.491	0.052	0	0	0.19	0.405	0.262	0.262	0.503	0.039	0.208	0	0.902	0.902	0.902	0.236	0.045	0
0	0.556	0.826	0.368	0.037	0.25	0.212	0.47	0.409	0.258	0.574	0.024	0.241	0.133	0.935	0.935	0.935	0.3	0.093	0.03
0	0.478	0.46	0.052	0	0	0.208	0.458	0.292	0.292	0.294	0.043	0.229	0	0.926	0.926	0.926	0.245	0.038	0
0	0.532	0.65	0.039	0	0	0.214	0.464	0.25	0.25	0.543	0.022	0.151	0	0.916	0.916	0.916	0.258	0.048	0
0	0.499	0.816	0.446	0.336	0.25	0.355	0.645	0.581	0.548	0.829	0.024	0.688	0.562	0.964	0.964	0.964	0.714	0.55	0.5

0	0.445	0.568	0.056	0	0	0.172	0.483	0.379	0.379	0.514	0.024	0.139	0	0.926	0.926	0.926	0.271	0.045	0
0	0.511	0.658	0.142	0	0.25	0.172	0.448	0.345	0.276	0.515	0.028	0.254	0	0.943	0.943	0.943	0.318	0.038	0
0	0.616	0.817	0.198	0.058	0.333	0.261	0.543	0.37	0.283	0.546	0.024	0.196	0	0.948	0.948	0.948	0.324	0.084	0
0	0.586	0.686	0.149	0	0.667	0.217	0.457	0.326	0.239	0.613	0.026	0.268	0	0.942	0.942	0.942	0.333	0.045	0
0	0.498	0.612	0	0	0.25	0.143	0.518	0.411	0.446	0.676	0.033	0.429	0.143	0.949	0.949	0.949	0.429	0	0
0	0.575	0.775	0.296	0.11	0.625	0.286	0.518	0.375	0.304	0.547	0.018	0.27	0.067	0.937	0.937	0.937	0.279	0.037	0
0	0.556	0.672	0.121	0	0.333	0.227	0.455	0.318	0.273	0.585	0.021	0.268	0	0.927	0.927	0.927	0.311	0	0
0	0.512	0.813	0.232	0.194	0.5	0.308	0.615	0.462	0.462	0.762	0.02	0.571	0.5	0.979	0.979	0.979	0.75	0.5	0.5
0	0.606	0.854	0.444	0.412	0.25	0.423	0.654	0.577	0.577	0.781	0.025	0.643	0.5	0.982	0.982	0.982	0.786	0.5	0.5
0	0.532	0.657	0.061	0	0	0.154	0.442	0.365	0.365	0.471	0.026	0.196	0	0.943	0.943	0.943	0.388	0.05	0
0	0.579	0.67	0.077	0	0	0.25	0.464	0.25	0.214	0.593	0.018	0.223	0	0.926	0.926	0.926	0.239	0.081	0
0	0.429	0.707	0.169	0	0	0.16	0.54	0.46	0.46	0.532	0.04	0.286	0	0.958	0.958	0.958	0.375	0.1	0
0	0.573	0.844	0.422	0.377	0.333	0.4	0.64	0.56	0.56	0.698	0.026	0.611	0.5	0.971	0.971	0.971	0.688	0.5	0.5
0	0.574	0.586	0.08	0	0.5	0.357	0.464	0.179	0.179	0.556	0.033	0.325	0	0.901	0.901	0.901	0.279	0.071	0
0	0.617	0.724	0.198	0	0	0.143	0.411	0.339	0.196	0.39	0.019	0.151	0	0.913	0.913	0.913	0.258	0.033	0
0	0.554	0.7	0	0	0	0.136	0.5	0.409	0.455	0.639	0.032	0.25	0	0.952	0.952	0.952	0.571	0	0
0	0.553	0.668	0.149	0	0	0.231	0.5	0.308	0.269	0.643	0.028	0.33	0.062	0.924	0.924	0.924	0.347	0.038	0
0	0.781	0.81	0.692	0.667	0.5	0.786	0.786	0.714	0.714	0.8	0.016	0.7	0.5	0.943	0.943	0.943	0.667	0.5	0.5
0	0.525	0.754	0.469	0.382	0.5	0.429	0.679	0.536	0.536	0.806	0.019	0.625	0.5	0.961	0.961	0.961	0.65	0.5	0.5
0	0.44	0.626	0.09	0	0	0.179	0.482	0.304	0.304	0.562	0.026	0.146	0	0.948	0.948	0.948	0.247	0.036	0
0	0.468	0.496	0.098	0	0	0.167	0.375	0.292	0.292	0.294	0.061	0.111	0	0.475	0.475	0.475	0.222	0.038	0
0	0.622	0.774	0.435	0.211	0.333	0.381	0.476	0.381	0.286	0.653	0.018	0.458	0.333	0.941	0.941	0.941	0.441	0.364	0.333
0	0.559	0.653	0.267	0.118	0.5	0.273	0.523	0.341	0.295	0.6	0.029	0.286	0.071	0.95	0.95	0.95	0.414	0.056	0
0	0.467	0.535	0.046	0	0	0.185	0.426	0.315	0.315	0.517	0.024	0.196	0	0.933	0.933	0.933	0.343	0.05	0
0	0.6	0.716	0.076	0	0	0.133	0.533	0.467	0.467	0.667	0.03	0.333	0	0.945	0.945	0.945	0.429	0.125	0
0	0.494	0.565	0.052	0	0	0.179	0.464	0.321	0.321	0.561	0.022	0.238	0	0.923	0.923	0.923	0.339	0.071	0
0	0.508	0.671	0.199	0.054	0.25	0.286	0.518	0.411	0.339	0.671	0.025	0.286	0.143	0.934	0.934	0.934	0.5	0.111	0
0	0.476	0.596	0.107	0	0.5	0.125	0.438	0.312	0.271	0.546	0.026	0.214	0	0.917	0.917	0.917	0.256	0.05	0
0	0.659	0.844	0.409	0.311	0.333	0.2	0.5	0.42	0.38	0.479	0.031	0.333	0.333	0.937	0.937	0.937	0.433	0.333	0.333
0	0.496	0.592	0.14	0	0	0.172	0.552	0.483	0.448	0.524	0.03	0.286	0	0.947	0.947	0.947	0.444	0.2	0
0	0.492	0.752	0.171	0.048	0.333	0.111	0.389	0.352	0.278	0.468	0.036	0.25	0.062	0.952	0.952	0.952	0.292	0.091	0.05
0	0.565	0.588	0.177	0	0	0.222	0.5	0.352	0.278	0.556	0.024	0.268	0	0.939	0.939	0.939	0.361	0.106	0
0	0.459	0.523	0.055	0	0	0.105	0.289	0.184	0.184	0.446	0.036	0.056	0	0.907	0.907	0.907	0.202	0.045	0
0	0.481	0.646	0.286	0.065	0.333	0.167	0.25	0.361	0.194	0.295	0.025	0.143	0.048	0.822	0.822	0.822	0.095	0.048	0
0	0.506	0.489	0.071	0	0.333	0.263	0.526	0.316	0.316	0.4	0.027	0.2	0	0.601	0.601	0.601	0.222	0	0
0	0.581	0.797	0.209	0.055	0	0.208	0.479	0.354	0.271	0.516	0.035	0.183	0	0.968	0.968	0.968	0.514	0.122	0.042
0	0.568	0.641	0.104	0	0.333	0.208	0.479	0.271	0.229	0.502	0.029	0.137	0	0.923	0.923	0.923	0.315	0.033	0
0	0.443	0.529	0.209	0	0.25	0.226	0.532	0.403	0.339	0.551	0.021	0.214	0.071	0.918	0.918	0.918	0.261	0.056	0
0	0.568	0.776	0.12	0	0.25	0.188	0.5	0.344	0.281	0.558	0.025	0.143	0	0.927	0.927	0.927	0.243	0.067	0
0	0.552	0.828	0.288	0	0	0.231	0.462	0.462	0.308	0.531	0.039	0.254	0.062	0.955	0.955	0.955	0.338	0.094	0
0	0.504	0.548	0.111	0	0.25	0.125	0.417	0.333	0.292	0.46	0.024	0.155	0	0.906	0.906	0.906	0.236	0.045	0
0	0.579	0.783	0.044	0	0.25	0.2	0.467	0.3	0.3	0.537	0.022	0.268	0	0.929	0.929	0.929	0.324	0	0
0	0.444	0.61	0.049	0	0	0.185	0.463	0.315	0.315	0.526	0.026	0.208	0	0.922	0.922	0.922	0.261	0.045	0
0	0.593	0.728	0.129	0	0	0.259	0.5	0.315	0.278	0.482	0.031	0.221	0	0.934	0.934	0.934	0.412	0.033	0
0	0.608	0.741	0.215	0	0.2	0.216	0.486	0.378	0.243	0.352	0.017	0.114	0	0.559	0.559	0.559	0.216	0.055	0
0	0.564	0.683	0.164	0	0.333	0.182	0.341	0.295	0.205	0.508	0.029	0.183	0	0.936	0.936	0.936	0.388	0.05	0
0	0.618	0.763	0.171	0	0	0.174	0.413	0.326	0.196	0.557	0.026	0.275	0	0.946	0.946	0.946	0.247	0.045	0
0	0.541	0.752	0.167	0	0	0.333	0.562	0.396	0.312	0.578	0.027	0.402	0	0.938	0.938	0.938	0.381	0.095	0
0	0.668	0.862	0.281	0.112	0.375	0.241	0.552	0.483	0.414	0.661	0.027	0.371	0.1	0.949	0.949	0.949	0.5	0.212	0.1
0	0.599	0.798	0.132	0.051	0	0.24	0.4	0.24	0.24	0.391	0.024	0.204	0.111	0.911	0.911	0.911	0.286	0.067	0.037
0	0.609	0.665	0.294	0.096	0.333	0.25	0.458	0.417	0.292	0.515	0.026	0.205	0	0.94	0.94	0.94	0.317	0.1	0.056
0	0.532	0.624	0	0	0	0.176	0.324	0.088	0.147	0.493	0.041	0.193	0	0.926	0.926	0.926	0.311	0	0
0	0.499	0.511	0.149	0	0.5	0.222	0.444	0.222	0.185	0.531	0.025	0.192	0	0.891	0.891	0.891	0.216	0.026	0
0	0.452	0.561	0.08	0	0	0.12	0.48	0.44	0.44	0.601	0.044	0.375	0	0.932	0.932	0.932	0.333	0.091	0
0	0.67	0.837	0.444	0.369	0.333	0.5	0.708	0.583	0.542	0.869	0.031	0.75	0.583	0.983	0.983	0.983	0.75	0.562	0.5
0	0.642	0.857	0.629	0.436	0.375	0.448	0.655	0.621	0.552	0.687	0.015	0.511	0.4	0.944	0.944	0.944	0.54	0.381	0.333
0	0.549	0.703	0.143	0.042	0.25	0.231	0.385	0.192	0.115	0.408	0.028	0.162	0	0.7	0.7	0.7	0.199	0.023	0
0	0.419	0.401	0.067	0	0	0.16	0.48	0.4	0.4	0.19	0.033	0.074	0	0.313	0.313	0.313	0.148	0.026	0
0	0.516	0.842	0.431	0.396	0.25	0.36	0.66	0.58	0.58	0.806	0.026	0.625	0.5	0.951	0.951	0.951	0.625	0.5	0.5
0	0.525	0.746	0.214	0.063	0	0.179	0.518	0.482	0.411	0.576	0.028	0.25	0.167	0.925	0.925	0.925	0.341	0.167	0.1
0	0.579	0.379	0	0	0.333	0.182	0.568	0.432	0.477	0.467	0.033	0.2	0	0.923	0.923	0.923	0.25	0	0
0	0.649	0.861	0.611	0.455	0.625	0.393	0.643	0.643	0.607	0.821	0.021	0.714	0.5	0.967	0.967	0.967	0.667	0.611	0.562

0	0.648	0.679	0.504	0.471	0.333	0.526	0.632	0.579	0.579	0.572	0.027	0.45	0.4	0.756	0.756	0.756	0.436	0.4	0.4
0	0.483	0.616	0.047	0	0	0.2	0.5	0.333	0.333	0.565	0.021	0.259	0	0.936	0.936	0.936	0.278	0.038	0
0	0.532	0.777	0.166	0	0.625	0.219	0.531	0.312	0.25	0.468	0.025	0.25	0.056	0.916	0.916	0.916	0.243	0.067	0
0	0.54	0.497	0.081	0	0	0.143	0.5	0.405	0.405	0.565	0.044	0.125	0	0.934	0.934	0.934	0.4	0.077	0
0	0.614	0.692	0	0	0.333	0.238	0.476	0.19	0.238	0.592	0.046	0.196	0	0.95	0.95	0.95	0.333	0	0
0	0.598	0.812	0.156	0.058	0	0.136	0.386	0.295	0.25	0.485	0.038	0.183	0	0.951	0.951	0.951	0.467	0.077	0.042
0	0.595	0.791	0.167	0	0	0.333	0.562	0.396	0.312	0.651	0.027	0.402	0.062	0.939	0.939	0.939	0.381	0.05	0
0	0.569	0.74	0.123	0	0.333	0.222	0.426	0.241	0.204	0.451	0.031	0.163	0.091	0.921	0.921	0.921	0.246	0.026	0
0	0.57	0.689	0.19	0	0.25	0.207	0.534	0.362	0.328	0.578	0.024	0.229	0	0.932	0.932	0.932	0.324	0.112	0
0	0.533	0.664	0.056	0	0.333	0.158	0.421	0.263	0.263	0.533	0.027	0.146	0	0.924	0.924	0.924	0.325	0	0
0	0.468	0.613	0.219	0.052	0.333	0.208	0.354	0.312	0.229	0.338	0.031	0.143	0	0.892	0.892	0.892	0.176	0.037	0
0	0.658	0.86	0.244	0	0.333	0.238	0.452	0.31	0.119	0.55	0.018	0.181	0	0.929	0.929	0.929	0.292	0.067	0
0	0.786	0.859	0.652	0.577	0.333	0.556	0.722	0.667	0.667	0.919	0.026	0.786	0.714	0.994	0.994	0.994	0.9	0.625	0.571
0	0.585	0.836	0.435	0.236	0.25	0.231	0.538	0.538	0.462	0.54	0.019	0.393	0.25	0.96	0.96	0.96	0.45	0.3	0.181
0	0.522	0.789	0.081	0	0	0.133	0.433	0.233	0.233	0.303	0.049	0.15	0.05	0.913	0.913	0.913	0.25	0.038	0
0	0.557	0.721	0.126	0.076	0.25	0.129	0.5	0.468	0.468	0.601	0.033	0.375	0	0.956	0.956	0.956	0.444	0.182	0.1
0	0.553	0.807	0.32	0.051	0.375	0.172	0.5	0.431	0.328	0.548	0.022	0.268	0	0.937	0.937	0.937	0.325	0.1	0
0	0.611	0.886	0.232	0	0	0.269	0.519	0.404	0.25	0.545	0.031	0.293	0	0.974	0.974	0.974	0.338	0.083	0
0	0.581	0.625	0	0	0	0.125	0.542	0.417	0.458	0.663	0.04	0.286	0	0.958	0.958	0.958	0.375	0	0
0	0.492	0.75	0.11	0	0.25	0.156	0.484	0.422	0.391	0.516	0.021	0.214	0	0.929	0.929	0.929	0.295	0.062	0
0	0.556	0.762	0.236	0.047	0.5	0.25	0.531	0.375	0.344	0.645	0.027	0.339	0.071	0.952	0.952	0.952	0.468	0.094	0
0	0.561	0.645	0.111	0	0.667	0.25	0.521	0.312	0.271	0.522	0.035	0.196	0	0.938	0.938	0.938	0.35	0.042	0
0	0.465	0.614	0.16	0.059	0.375	0.172	0.466	0.397	0.397	0.498	0.024	0.143	0	0.924	0.924	0.924	0.278	0	0
0	0.521	0.609	0.063	0	0	0.188	0.312	0.125	0.125	0.498	0.05	0.134	0	0.925	0.925	0.925	0.282	0.042	0
0	0.544	0.777	0.133	0	0	0.161	0.468	0.371	0.306	0.54	0.028	0.125	0	0.934	0.934	0.934	0.312	0.038	0
0	0.503	0.667	0.051	0	0	0.182	0.455	0.182	0.182	0.591	0.033	0.222	0	0.938	0.938	0.938	0.35	0	0
0	0.559	0.702	0.094	0.05	0.5	0.269	0.481	0.25	0.25	0.528	0.015	0.204	0	0.922	0.922	0.922	0.312	0	0
0	0.443	0.538	0.062	0	0	0.211	0.395	0.237	0.237	0.468	0.026	0.222	0	0.916	0.916	0.916	0.188	0	0
0	0.585	0.723	0.211	0	0.25	0.182	0.455	0.303	0.182	0.583	0.027	0.25	0.083	0.953	0.953	0.953	0.424	0.061	0
0	0.575	0.706	0.506	0.421	0.5	0.44	0.66	0.58	0.58	0.724	0.019	0.571	0.5	0.956	0.956	0.956	0.625	0.5	0.5
0	0.524	0.754	0.143	0	0	0.125	0.312	0.25	0.188	0.354	0.049	0.188	0.125	0.915	0.915	0.915	0.2	0.056	0
0	0.644	0.869	0.307	0	0.5	0.174	0.435	0.348	0.13	0.415	0.029	0.167	0.125	0.893	0.893	0.893	0.356	0.056	0
0	0.553	0.561	0.122	0.049	0.333	0.13	0.37	0.152	0.152	0.495	0.025	0.213	0	0.919	0.919	0.919	0.217	0	0
0	0.59	0.868	0.229	0.052	0	0.258	0.581	0.452	0.355	0.592	0.022	0.236	0	0.974	0.974	0.974	0.389	0.122	0.045
0	0.659	0.818	0.292	0	0	0.31	0.5	0.362	0.19	0.591	0.033	0.26	0.03	0.94	0.94	0.94	0.304	0.042	0
0	0.491	0.472	0.111	0	0	0.208	0.479	0.354	0.312	0.475	0.028	0.155	0	0.901	0.901	0.901	0.234	0.062	0
0	0.555	0.687	0.211	0	0.5	0.304	0.5	0.37	0.239	0.52	0.029	0.125	0	0.944	0.944	0.944	0.449	0.045	0
0	0.572	0.696	0.058	0	0	0.208	0.458	0.292	0.292	0.537	0.046	0.134	0	0.929	0.929	0.929	0.278	0.042	0
0	0.636	0.766	0.445	0.327	0.25	0.414	0.621	0.552	0.483	0.767	0.03	0.6	0.5	0.975	0.975	0.975	0.625	0.538	0.5
0	0.473	0.602	0.155	0	0	0.214	0.464	0.393	0.321	0.588	0.024	0.268	0.071	0.929	0.929	0.929	0.267	0.094	0
0	0.487	0.426	0.046	0	0.5	0.2	0.44	0.28	0.28	0.546	0.026	0.167	0	0.924	0.924	0.924	0.306	0	0
0	0.507	0.718	0.413	0.374	0.2	0.379	0.621	0.552	0.552	0.77	0.024	0.583	0.5	0.951	0.951	0.951	0.625	0.5	0.5
0	0.568	0.701	0.09	0	0.25	0.233	0.55	0.383	0.35	0.61	0.024	0.31	0	0.941	0.941	0.941	0.367	0.062	0
0	0.475	0.526	0	0	0	0.158	0.526	0.421	0.474	0.6	0.025	0.4	0	0.929	0.929	0.929	0.286	0	0
0	0.46	0.663	0	0	0	0.192	0.481	0.288	0.327	0.598	0.021	0.343	0	0.935	0.935	0.935	0.386	0	0
0	0.58	0.657	0.214	0	0.333	0.208	0.5	0.417	0.292	0.513	0.031	0.259	0	0.952	0.952	0.952	0.388	0.1	0
0	0.487	0.547	0.144	0	0.333	0.25	0.5	0.3	0.2	0.655	0.036	0.268	0	0.94	0.94	0.94	0.367	0.088	0
0	0.56	0.671	0.144	0.052	0.25	0.214	0.5	0.357	0.321	0.637	0.027	0.411	0.143	0.95	0.95	0.95	0.45	0.111	0.062
0	0.599	0.396	0.126	0.076	0.25	0.172	0.534	0.466	0.466	0.658	0.024	0.375	0.125	0.946	0.946	0.946	0.333	0.222	0.125
0	0.596	0.816	0.089	0	0	0.217	0.457	0.239	0.196	0.622	0.033	0.229	0	0.954	0.954	0.954	0.479	0.077	0
0	0.42	0.438	0.082	0	0	0.214	0.429	0.321	0.286	0.515	0.031	0.236	0	0.933	0.933	0.933	0.3	0.033	0
0	0.55	0.656	0.165	0	0.333	0.19	0.405	0.31	0.214	0.518	0.033	0.196	0	0.922	0.922	0.922	0.303	0.084	0
0	0.605	0.613	0.177	0.071	0	0.185	0.537	0.426	0.426	0.49	0.036	0.25	0.125	0.946	0.946	0.946	0.364	0.056	0
0	0.648	0.837	0.613	0.458	0.5	0.393	0.589	0.625	0.554	0.521	0.014	0.467	0.4	0.92	0.92	0.92	0.407	0.37	0.333
0	0.633	0.764	0.516	0.465	0.625	0.407	0.63	0.556	0.593	0.775	0.021	0.625	0.5	0.96	0.96	0.96	0.611	0.5	0.5
0	0.496	0.613	0.085	0	0	0.167	0.521	0.438	0.438	0.54	0.032	0.143	0	0.954	0.954	0.954	0.5	0.1	0
0	0.505	0.653	0	0	0	0.111	0.444	0.167	0.222	0.575	0.039	0.238	0.083	0.929	0.929	0.929	0.357	0	0
0	0.673	0.76	0.361	0.327	0.25	0.448	0.655	0.483	0.483	0.693	0.017	0.444	0.333	0.941	0.941	0.941	0.524	0.333	0.333
0	0.544	0.473	0.096	0	0.5	0.167	0.438	0.312	0.271	0.515	0.031	0.134	0	0.927	0.927	0.927	0.336	0.042	0
0	0.528	0.709	0.049	0	0	0.227	0.409	0.227	0.227	0.493	0.043	0.139	0	0.926	0.926	0.926	0.24	0.042	0
0	0.566	0.682	0.199	0	0.5	0.233	0.567	0.4	0.333	0.633	0.016	0.25	0	0.933	0.933	0.933	0.343	0.042	0
0	0.575	0.745	0	0	0	0.143	0.476	0.381	0.429	0.524	0.04	0.143	0	0.964	0.964	0.964	0.571	0	0

0	0.558	0.716	0.218	0	0.25	0.24	0.52	0.4	0.28	0.561	0.026	0.267	0.071	0.931	0.931	0.931	0.347	0.153	0
0	0.488	0.698	0.098	0	0.5	0.179	0.518	0.375	0.339	0.624	0.027	0.259	0.062	0.943	0.943	0.943	0.438	0.095	0
0	0.508	0.684	0.177	0.063	0.5	0.259	0.537	0.426	0.389	0.651	0.032	0.402	0.062	0.944	0.944	0.944	0.4	0.05	0
0	0.646	0.813	0.231	0	0.333	0.28	0.48	0.36	0.24	0.547	0.029	0.214	0.083	0.941	0.941	0.941	0.359	0.037	0
0	0.523	0.672	0.169	0	0.5	0.15	0.35	0.25	0.2	0.51	0.026	0.205	0.125	0.932	0.932	0.932	0.286	0.056	0
0	0.609	0.764	0.083	0	0	0.208	0.458	0.167	0.125	0.54	0.028	0.173	0.042	0.942	0.942	0.942	0.322	0.03	0
0	0.467	0.673	0.149	0	0	0.25	0.554	0.411	0.339	0.643	0.026	0.357	0	0.948	0.948	0.948	0.5	0.106	0
0	0.651	0.653	0.491	0.342	0.667	0.522	0.587	0.5	0.457	0.634	0.018	0.437	0.333	0.91	0.91	0.91	0.412	0.333	0.333
0	0.562	0.793	0.401	0.054	0.5	0.28	0.56	0.48	0.28	0.389	0.037	0.268	0.173	0.919	0.919	0.919	0.315	0.141	0
0	0.431	0.648	0.198	0	0	0.167	0.472	0.472	0.417	0.562	0.044	0.143	0	0.959	0.959	0.959	0.571	0.25	0
0	0.545	0.684	0.13	0	0.25	0.258	0.548	0.387	0.323	0.512	0.027	0.143	0	0.937	0.937	0.937	0.325	0.045	0
0	0.534	0.77	0.131	0.048	0.5	0.286	0.482	0.304	0.268	0.39	0.013	0.19	0.048	0.904	0.904	0.904	0.255	0	0
0	0.564	0.743	0.147	0	0.5	0.296	0.481	0.333	0.259	0.532	0.018	0.283	0.133	0.915	0.915	0.915	0.261	0.089	0
0	0.651	0.738	0.415	0.315	0.25	0.267	0.483	0.417	0.417	0.525	0.021	0.429	0.333	0.934	0.934	0.934	0.444	0.333	0.333
0	0.594	0.83	0.336	0	0	0.292	0.521	0.438	0.271	0.571	0.038	0.214	0	0.966	0.966	0.966	0.436	0.136	0
0	0.571	0.731	0.294	0.056	0	0.238	0.476	0.333	0.238	0.532	0.039	0.238	0.056	0.95	0.95	0.95	0.317	0.145	0.056
0	0.572	0.75	0.48	0.333	0.25	0.13	0.478	0.435	0.391	0.361	0.019	0.292	0.25	0.678	0.678	0.678	0.312	0.25	0.25
0	0.421	0.434	0.121	0	0.25	0.161	0.484	0.387	0.355	0.48	0.025	0.214	0.071	0.913	0.913	0.913	0.275	0.056	0
0	0.554	0.566	0	0	0	0.2	0.433	0.1	0.167	0.324	0.066	0.111	0	0.904	0.904	0.904	0.238	0	0
0	0.516	0.621	0.122	0	0	0.231	0.519	0.365	0.365	0.523	0.021	0.243	0	0.913	0.913	0.913	0.211	0	0
0	0.518	0.656	0.098	0	0	0.296	0.574	0.352	0.315	0.692	0.03	0.339	0.196	0.944	0.944	0.944	0.375	0.106	0
0	0.677	0.774	0.545	0.467	0.333	0.478	0.674	0.587	0.587	0.714	0.024	0.571	0.5	0.959	0.959	0.959	0.714	0.5	0.5
0	0.631	0.861	0.236	0.051	0	0.31	0.603	0.431	0.328	0.628	0.022	0.238	0	0.951	0.951	0.951	0.589	0.111	0.062
0	0.483	0.453	0	0	0.375	0.148	0.556	0.444	0.481	0.633	0.02	0.4	0	0.927	0.927	0.927	0.375	0	0
0	0.528	0.652	0.094	0	0	0.222	0.537	0.315	0.315	0.566	0.024	0.196	0	0.932	0.932	0.932	0.354	0	0
0	0.47	0.635	0.046	0	0	0.217	0.457	0.239	0.239	0.565	0.031	0.188	0	0.933	0.933	0.933	0.35	0.042	0
0	0.461	0.579	0.093	0	0	0.154	0.558	0.481	0.481	0.6	0.025	0.4	0	0.939	0.939	0.939	0.429	0.143	0
0	0.626	0.795	0	0	0.333	0.227	0.455	0.182	0.227	0.521	0.023	0.125	0	0.943	0.943	0.943	0.35	0	0
0	0.502	0.531	0.055	0	0	0.2	0.5	0.34	0.34	0.533	0.031	0.196	0.125	0.918	0.918	0.918	0.278	0.056	0
0	0.548	0.777	0	0	0	0.259	0.444	0.222	0.259	0.549	0.032	0.188	0.188	0.935	0.935	0.935	0.298	0	0
0	0.554	0.599	0.128	0	0	0.217	0.37	0.196	0.152	0.455	0.032	0.179	0	0.908	0.908	0.908	0.348	0.03	0
0	0.421	0.621	0.116	0	0.25	0.167	0.417	0.333	0.292	0.502	0.022	0.205	0	0.935	0.935	0.935	0.222	0.056	0
0	0.679	0.884	0.653	0.517	0.5	0.273	0.568	0.614	0.568	0.565	0.017	0.4	0.333	0.947	0.947	0.947	0.444	0.381	0.333
0	0.558	0.63	0.238	0	0.333	0.2	0.46	0.38	0.26	0.554	0.027	0.196	0	0.943	0.943	0.943	0.45	0.042	0
0	0.515	0.573	0.152	0.055	0.333	0.185	0.444	0.37	0.333	0.552	0.029	0.243	0	0.943	0.943	0.943	0.183	0.042	0
0	0.512	0.697	0.248	0	0	0.222	0.519	0.444	0.333	0.595	0.021	0.306	0	0.918	0.918	0.918	0.367	0.177	0
0	0.704	0.896	0.479	0.364	0.667	0.44	0.6	0.52	0.44	0.631	0.017	0.429	0.333	0.94	0.94	0.94	0.467	0.361	0.333
0	0.6	0.827	0.052	0	0.667	0.19	0.452	0.214	0.214	0.536	0.027	0.188	0.125	0.945	0.945	0.945	0.417	0.038	0
0	0.591	0.756	0.158	0	0	0.2	0.5	0.34	0.26	0.556	0.027	0.25	0	0.927	0.927	0.927	0.333	0.091	0
0	0.531	0.691	0.304	0	0	0.211	0.395	0.395	0.184	0.501	0.05	0.216	0	0.942	0.942	0.942	0.372	0.075	0
0	0.614	0.8	0.11	0	0.25	0.207	0.431	0.224	0.19	0.527	0.021	0.214	0.125	0.931	0.931	0.931	0.378	0.033	0
0	0.568	0.754	0.086	0	0	0.2	0.46	0.22	0.18	0.535	0.036	0.174	0.037	0.928	0.928	0.928	0.285	0.033	0
0	0.537	0.602	0.196	0.047	0.333	0.24	0.42	0.26	0.18	0.541	0.022	0.227	0.083	0.937	0.937	0.937	0.267	0.054	0
0	0.593	0.775	0.202	0.12	0.333	0.182	0.455	0.364	0.318	0.558	0.033	0.312	0.125	0.95	0.95	0.95	0.247	0.167	0.125
0	0.637	0.825	0.218	0.052	0.25	0.286	0.571	0.393	0.321	0.6	0.02	0.278	0.056	0.963	0.963	0.963	0.586	0.139	0.042
0	0.509	0.703	0.164	0	0	0.182	0.5	0.409	0.348	0.526	0.026	0.196	0	0.935	0.935	0.935	0.381	0.092	0
0	0.6	0.854	0.275	0.144	0	0.345	0.569	0.431	0.397	0.625	0.031	0.247	0.111	0.975	0.975	0.975	0.583	0.182	0.15
0	0.62	0.827	0.48	0.412	0.375	0.385	0.654	0.577	0.577	0.742	0.016	0.6	0.5	0.957	0.957	0.957	0.667	0.5	0.5
0	0.626	0.801	0.559	0.501	0.833	0.632	0.684	0.579	0.579	0.509	0.019	0.417	0.333	0.644	0.644	0.644	0.4	0.333	0.333
0	0.532	0.475	0.134	0	0	0.172	0.534	0.466	0.431	0.55	0.024	0.25	0	0.934	0.934	0.934	0.273	0.1	0
0	0.619	0.798	0.321	0.047	0.333	0.25	0.479	0.312	0.188	0.575	0.025	0.278	0.097	0.928	0.928	0.928	0.226	0.102	0
0	0.52	0.735	0.239	0.069	0.333	0.19	0.476	0.429	0.333	0.625	0.022	0.188	0	0.932	0.932	0.932	0.303	0.062	0
0	0.599	0.582	0.134	0	0.5	0.176	0.412	0.235	0.176	0.501	0.041	0.125	0	0.917	0.917	0.917	0.277	0.042	0
0	0.496	0.667	0.123	0	0	0.238	0.452	0.357	0.31	0.531	0.024	0.214	0	0.929	0.929	0.929	0.375	0.083	0
0	0.55	0.793	0.047	0	0.5	0.192	0.481	0.288	0.288	0.637	0.028	0.292	0.083	0.946	0.946	0.946	0.422	0	0
0	0.4	0.505	0.136	0	0	0.174	0.5	0.413	0.37	0.281	0.03	0.196	0	0.918	0.918	0.918	0.211	0.112	0
0	0.559	0.691	0.211	0	0	0.24	0.38	0.18	0.1	0.406	0.025	0.189	0.083	0.69	0.69	0.69	0.225	0.047	0
0	0.564	0.815	0.101	0.058	0.333	0.222	0.5	0.352	0.352	0.51	0.029	0.234	0	0.944	0.944	0.944	0.359	0	0
0	0.531	0.588	0.093	0	0.333	0.208	0.458	0.292	0.25	0.52	0.029	0.125	0	0.939	0.939	0.939	0.382	0.042	0
0	0.614	0.777	0.133	0.05	0	0.31	0.552	0.345	0.31	0.48	0.024	0.183	0	0.951	0.951	0.951	0.467	0.077	0.042
0	0.551	0.634	0.056	0	0	0.115	0.462	0.346	0.346	0.491	0.039	0.118	0	0.92	0.92	0.92	0.243	0.062	0
0	0.481	0.59	0.1	0	0	0.2	0.467	0.367	0.333	0.545	0.03	0.292	0	0.933	0.933	0.933	0.333	0.045	0

0	0.496	0.647	0	0	0	0.16	0.4	0.2	0.24	0.358	0.04	0.131	0	0.893	0.893	0.893	0.174	0	0
0	0.598	0.826	0.239	0	0	0.227	0.455	0.318	0.182	0.47	0.031	0.217	0.108	0.907	0.907	0.907	0.311	0.033	0
0	0.462	0.254	0	0	0	0.095	0.476	0.381	0.429	0.524	0.025	0.143	0	0.929	0.929	0.929	0.3	0	0
0	0.631	0.728	0.271	0	0.5	0.333	0.5	0.357	0.214	0.613	0.026	0.205	0.071	0.934	0.934	0.934	0.325	0.142	0
0	0.569	0.746	0.1	0	0	0.1	0.3	0.2	0.15	0.526	0.036	0.217	0.056	0.953	0.953	0.953	0.361	0.045	0
0	0.595	0.758	0.213	0	0.5	0.375	0.521	0.312	0.146	0.631	0.029	0.351	0.048	0.945	0.945	0.945	0.429	0.065	0
0	0.621	0.881	0.307	0.14	0.625	0.345	0.552	0.379	0.345	0.623	0.016	0.319	0.125	0.945	0.945	0.945	0.369	0.111	0.056
0	0.524	0.667	0	0	0	0.148	0.5	0.352	0.389	0.442	0.024	0.188	0	0.905	0.905	0.905	0.221	0	0
0	0.489	0.25	0	0	0	0.067	0.483	0.45	0.483	0.448	0.025	0.143	0	0.915	0.915	0.915	0.182	0	0
0	0.618	0.843	0.485	0.399	0.333	0.375	0.625	0.583	0.542	0.714	0.033	0.5	0.5	0.969	0.969	0.969	0.667	0.5	0.5
0	0.595	0.779	0.081	0	0	0.25	0.464	0.25	0.214	0.466	0.02	0.103	0	0.92	0.92	0.92	0.317	0.067	0
0	0.711	0.787	0.497	0.381	0.25	0.5	0.696	0.518	0.518	0.694	0.016	0.467	0.333	0.926	0.926	0.926	0.556	0.333	0.333
0	0.624	0.756	0.105	0.056	0.375	0.2	0.42	0.3	0.3	0.578	0.028	0.205	0	0.937	0.937	0.937	0.389	0.05	0
0	0.495	0.584	0.196	0.057	0.333	0.19	0.405	0.31	0.214	0.38	0.025	0.167	0	0.883	0.883	0.883	0.188	0.078	0
0	0.487	0.434	0	0	0	0.133	0.467	0.333	0.367	0.51	0.021	0.205	0	0.921	0.921	0.921	0.243	0	0
0	0.433	0.331	0	0	0	0.143	0.429	0.321	0.357	0.496	0.022	0.205	0	0.897	0.897	0.897	0.236	0	0
0	0.583	0.722	0.135	0	0	0.25	0.5	0.417	0.417	0.514	0.061	0.333	0.167	0.958	0.958	0.958	0.333	0.167	0
0	0.503	0.602	0.178	0.052	0.25	0.148	0.444	0.333	0.296	0.52	0.032	0.188	0	0.925	0.925	0.925	0.25	0.1	0.05
0	0.486	0.57	0.11	0	0.25	0.179	0.518	0.411	0.375	0.603	0.028	0.31	0	0.908	0.908	0.908	0.25	0.062	0
0	0.603	0.709	0.075	0	0.5	0.25	0.406	0.219	0.219	0.509	0.041	0.25	0	0.93	0.93	0.93	0.303	0	0
0	0.437	0.537	0	0	0	0.118	0.265	0.088	0.147	0.409	0.05	0.062	0	0.927	0.927	0.927	0.227	0	0
0	0.612	0.851	0.534	0.431	0.333	0.417	0.625	0.625	0.583	0.724	0.026	0.571	0.5	0.948	0.948	0.948	0.556	0.5	0.5
0	0.473	0.514	0.095	0	0	0.174	0.543	0.457	0.457	0.663	0.033	0.429	0.286	0.954	0.954	0.954	0.5	0.1	0
0	0.559	0.807	0.415	0.355	0.25	0.357	0.625	0.589	0.554	0.8	0.022	0.7	0.5	0.958	0.958	0.958	0.643	0.5	0.5
0	0.538	0.627	0.105	0	0	0.16	0.46	0.34	0.3	0.575	0.035	0.167	0.056	0.936	0.936	0.936	0.3	0.045	0
0	0.625	0.871	0.553	0.354	0.5	0.433	0.617	0.55	0.517	0.669	0.014	0.533	0.533	0.934	0.934	0.934	0.533	0.381	0.333
0	0.491	0.621	0.053	0	0	0.174	0.37	0.239	0.239	0.454	0.033	0.196	0	0.949	0.949	0.949	0.422	0.042	0
0	0.551	0.655	0.288	0.277	0.25	0.429	0.571	0.429	0.429	0.521	0.015	0.417	0.333	0.915	0.915	0.915	0.455	0.333	0.333
0	0.641	0.595	0.185	0	0.333	0.333	0.595	0.405	0.31	0.649	0.031	0.268	0.071	0.943	0.943	0.943	0.312	0.045	0
0	0.458	0.26	0	0	0	0.071	0.482	0.446	0.482	0.448	0.024	0.143	0	0.93	0.93	0.93	0.333	0	0
0	0.635	0.67	0	0	0	0.235	0.412	0.118	0.176	0.539	0.036	0.196	0	0.939	0.939	0.939	0.473	0	0
0	0.615	0.717	0.143	0	0	0.25	0.464	0.214	0.143	0.617	0.019	0.262	0	0.935	0.935	0.935	0.402	0.061	0
0	0.508	0.578	0.227	0.094	0.333	0.269	0.519	0.365	0.327	0.534	0.033	0.174	0	0.943	0.943	0.943	0.367	0.1	0.056
0	0.554	0.578	0.21	0.06	0.5	0.222	0.481	0.444	0.37	0.557	0.026	0.261	0.045	0.944	0.944	0.944	0.222	0.056	0
0	0.538	0.674	0.253	0.12	0.5	0.24	0.5	0.42	0.38	0.52	0.029	0.259	0	0.938	0.938	0.938	0.286	0.045	0
0	0.574	0.705	0.094	0	0	0.214	0.464	0.357	0.321	0.474	0.036	0.161	0	0.944	0.944	0.944	0.304	0.036	0
0	0.409	0.626	0.042	0	0	0.148	0.407	0.296	0.296	0.518	0.026	0.143	0	0.93	0.93	0.93	0.183	0.045	0
0	0.717	0.875	0.223	0.134	0.5	0.389	0.5	0.278	0.278	0.636	0.031	0.354	0	0.964	0.964	0.964	0.394	0.042	0
0	0.548	0.658	0.144	0	0.375	0.233	0.55	0.383	0.317	0.54	0.027	0.214	0	0.931	0.931	0.931	0.388	0.111	0
0	0.619	0.854	0.42	0.31	0.25	0.387	0.613	0.484	0.419	0.692	0.02	0.574	0.417	0.958	0.958	0.958	0.567	0.364	0.333
0	0.502	0.698	0.195	0.053	0.25	0.185	0.463	0.389	0.315	0.53	0.022	0.205	0	0.929	0.929	0.929	0.311	0.05	0
0	0.576	0.62	0.161	0.091	0.5	0.192	0.462	0.231	0.231	0.5	0.035	0.157	0	0.936	0.936	0.936	0.274	0	0
0	0.477	0.761	0.588	0.462	0.375	0.407	0.648	0.648	0.611	0.781	0.02	0.643	0.5	0.947	0.947	0.947	0.625	0.556	0.5
0	0.522	0.7	0.223	0	0	0.273	0.477	0.386	0.25	0.616	0.039	0.214	0	0.93	0.93	0.93	0.438	0.095	0
0	0.55	0.552	0.06	0	0.25	0.222	0.519	0.37	0.37	0.528	0.021	0.208	0	0.91	0.91	0.91	0.292	0	0
0	0.615	0.743	0.452	0.389	0.333	0.105	0.474	0.421	0.421	0.333	0.029	0.333	0.333	0.857	0.857	0.857	0.37	0.333	0.333
0	0.58	0.601	0.126	0.067	0.5	0.263	0.474	0.263	0.263	0.59	0.027	0.354	0	0.932	0.932	0.932	0.347	0.045	0
0	0.408	0.465	0.071	0	0.5	0.067	0.3	0.233	0.233	0.292	0.041	0.143	0	0.866	0.866	0.866	0.125	0.056	0
0	0.724	0.839	0.278	0.193	0.333	0.292	0.562	0.438	0.438	0.75	0.022	0.42	0.357	0.973	0.973	0.973	0.75	0.167	0.125
0	0.457	0.582	0.115	0	0.333	0.15	0.375	0.225	0.175	0.371	0.035	0.125	0	0.613	0.613	0.613	0.192	0.037	0
0	0.651	0.859	0.191	0.045	0.25	0.29	0.516	0.387	0.29	0.656	0.024	0.279	0.136	0.968	0.968	0.968	0.417	0.112	0.036
0	0.592	0.747	0.385	0	0	0.158	0.526	0.579	0.421	0.565	0.032	0.286	0	0.965	0.965	0.965	0.571	0.222	0
0	0.493	0.619	0	0	0.25	0.161	0.484	0.323	0.355	0.453	0.018	0.162	0	0.922	0.922	0.922	0.208	0	0
0	0.628	0.769	0.454	0.39	0.75	0.393	0.589	0.518	0.518	0.556	0.016	0.444	0.333	0.914	0.914	0.914	0.458	0.37	0.333
0	0.567	0.669	0.048	0	0.333	0.231	0.519	0.288	0.288	0.518	0.031	0.127	0	0.937	0.937	0.937	0.35	0	0
0	0.423	0.506	0.128	0	0	0.179	0.536	0.464	0.429	0.601	0.024	0.375	0	0.956	0.956	0.956	0.444	0.083	0
0	0.597	0.646	0.181	0	0.333	0.292	0.542	0.375	0.333	0.66	0.033	0.357	0.071	0.94	0.94	0.94	0.41	0.056	0
0	0.469	0.69	0.183	0.068	0	0.192	0.519	0.442	0.404	0.57	0.022	0.286	0.214	0.919	0.919	0.919	0.312	0.125	0.071
0	0.515	0.46	0	0	0	0.154	0.519	0.404	0.442	0.562	0.024	0.286	0	0.942	0.942	0.942	0.3	0	0
0	0.513	0.612	0	0	0	0.077	0.385	0.154	0.231	0.46	0.058	0.143	0	0.878	0.878	0.878	0.125	0	0
0	0.595	0.813	0.131	0	0	0.2	0.48	0.2	0.12	0.632	0.026	0.234	0.048	0.927	0.927	0.927	0.341	0.033	0
0	0.663	0.896	0.189	0.103	0.25	0.31	0.534	0.397	0.362	0.574	0.02	0.375	0.062	0.949	0.949	0.949	0.389	0.042	0

0	0.442	0.474	0	0	0	0.182	0.455	0.318	0.364	0.281	0.036	0.143	0	0.902	0.902	0.902	0.1	0	0
0	0.457	0.586	0	0	0.5	0.12	0.54	0.46	0.5	0.595	0.044	0.375	0.25	0.942	0.942	0.942	0.333	0	0
0	0.649	0.836	0.328	0.041	0.25	0.214	0.446	0.339	0.161	0.608	0.019	0.296	0.079	0.94	0.94	0.94	0.341	0.077	0.028
0	0.519	0.572	0.16	0.087	0.5	0.214	0.464	0.25	0.25	0.61	0.041	0.259	0.062	0.917	0.917	0.917	0.261	0.05	0
0	0.513	0.623	0.147	0.061	0	0.286	0.429	0.286	0.286	0.492	0.039	0.259	0.062	0.92	0.92	0.92	0.295	0.05	0
0	0.535	0.663	0.146	0	0.5	0.292	0.5	0.333	0.25	0.507	0.031	0.259	0	0.934	0.934	0.934	0.336	0.042	0
0	0.402	0.221	0.082	0	0	0.071	0.482	0.482	0.482	0.55	0.04	0.25	0	0.927	0.927	0.927	0.2	0.1	0
0	0.529	0.534	0	0	0.333	0.105	0.474	0.368	0.421	0.643	0.04	0.429	0	0.929	0.929	0.929	0.375	0	0
0	0.631	0.838	0.192	0	0.667	0.217	0.435	0.348	0.217	0.483	0.023	0.151	0.083	0.896	0.896	0.896	0.347	0.12	0
0	0.577	0.754	0.229	0	0.25	0.25	0.562	0.438	0.344	0.655	0.021	0.312	0.062	0.957	0.957	0.957	0.422	0.087	0
0	0.518	0.826	0.156	0	0.25	0.208	0.438	0.312	0.229	0.39	0.025	0.229	0.067	0.883	0.883	0.883	0.222	0.033	0
0	0.571	0.701	0.114	0	0.5	0.261	0.478	0.304	0.261	0.596	0.024	0.286	0	0.936	0.936	0.936	0.312	0.045	0
0	0.578	0.74	0.051	0	0.333	0.222	0.5	0.315	0.315	0.546	0.026	0.279	0	0.944	0.944	0.944	0.359	0	0
0	0.724	0.9	0.482	0.386	0.333	0.375	0.646	0.562	0.562	0.774	0.031	0.562	0.5	0.982	0.982	0.982	0.75	0.5	0.5
0	0.625	0.789	0.194	0	0	0.25	0.5	0.333	0.25	0.51	0.033	0.188	0	0.957	0.957	0.957	0.354	0.088	0
0	0.421	0.437	0.061	0	0	0.125	0.438	0.354	0.354	0.451	0.031	0.188	0.125	0.909	0.909	0.909	0.167	0.056	0
0	0.461	0.36	0	0	0	0.077	0.481	0.442	0.481	0.467	0.025	0	0	0.92	0.92	0.92	0.25	0	0
0	0.454	0.603	0.245	0.095	0	0.167	0.521	0.521	0.479	0.524	0.04	0.25	0	0.952	0.952	0.952	0.333	0.2	0.111
0	0.552	0.41	0.255	0.06	0.5	0.259	0.519	0.444	0.37	0.488	0.026	0.236	0	0.935	0.935	0.935	0.311	0.087	0
0	0.468	0.426	0.175	0	0.333	0.192	0.442	0.327	0.288	0.509	0.038	0.125	0	0.927	0.927	0.927	0.278	0.086	0
0	0.525	0.31	0	0	0	0.074	0.5	0.463	0.5	0.667	0.028	0.25	0	0.919	0.919	0.919	0.333	0	0
0	0.686	0.89	0.577	0.393	0.833	0.52	0.66	0.58	0.5	0.694	0.023	0.463	0.333	0.933	0.933	0.933	0.476	0.37	0.333
0	0.626	0.851	0.154	0	0.25	0.259	0.481	0.296	0.222	0.534	0.019	0.208	0.111	0.939	0.939	0.939	0.426	0.048	0
0	0.443	0.513	0	0	0	0.167	0.45	0.317	0.35	0.524	0.024	0.259	0	0.927	0.927	0.927	0.2	0	0
0	0.563	0.6	0.237	0.132	0.333	0.286	0.452	0.405	0.357	0.531	0.033	0.259	0.083	0.934	0.934	0.934	0.389	0.143	0.083
0	0.613	0.8	0.355	0.132	0.333	0.286	0.452	0.452	0.357	0.551	0.027	0.31	0	0.932	0.932	0.932	0.292	0.1	0.056
0	0.478	0.635	0.072	0	0	0.226	0.5	0.339	0.339	0.555	0.03	0.196	0	0.933	0.933	0.933	0.3	0.045	0
0	0.387	0.425	0	0	0	0.129	0.468	0.371	0.403	0.264	0.043	0.112	0	0.93	0.93	0.93	0.153	0	0
0	0.618	0.862	0.296	0.08	0	0.333	0.5	0.367	0.233	0.567	0.041	0.278	0.125	0.961	0.961	0.961	0.383	0.077	0.042
0	0.466	0.457	0.294	0.126	0	0.13	0.522	0.522	0.522	0.456	0.032	0.167	0.167	0.939	0.939	0.939	0.429	0.286	0.167
0	0.555	0.744	0.087	0	0.375	0.2	0.5	0.367	0.333	0.588	0.021	0.268	0.071	0.926	0.926	0.926	0.298	0.056	0
0	0.547	0.762	0.221	0	0	0.259	0.5	0.352	0.278	0.57	0.035	0.247	0.091	0.96	0.96	0.96	0.389	0.083	0
0	0.686	0.878	0.56	0.373	0.333	0.542	0.708	0.583	0.5	0.785	0.027	0.611	0.5	0.976	0.976	0.976	0.65	0.542	0.5
0	0.568	0.671	0.15	0	0.333	0.143	0.381	0.238	0.143	0.356	0.024	0.179	0	0.87	0.87	0.87	0.229	0.037	0
0	0.545	0.889	0.439	0.412	0.5	0.387	0.677	0.581	0.581	0.806	0.026	0.65	0.5	0.988	0.988	0.988	0.75	0.5	0.5
0	0.723	0.847	0.435	0.37	0.833	0.571	0.667	0.476	0.476	0.649	0.023	0.464	0.333	0.902	0.902	0.902	0.418	0.333	0.333
0	0.614	0.857	0.159	0	0.25	0.222	0.389	0.204	0.093	0.508	0.025	0.163	0	0.942	0.942	0.942	0.405	0.089	0
0	0.627	0.775	0.299	0.055	0.375	0.267	0.567	0.467	0.367	0.656	0.02	0.458	0.25	0.95	0.95	0.95	0.417	0.188	0.083
0	0.522	0.693	0.058	0	0	0.136	0.386	0.25	0.25	0.486	0.039	0.143	0	0.939	0.939	0.939	0.367	0.042	0
0	0.493	0.575	0.204	0	0	0.12	0.52	0.52	0.48	0.578	0.025	0.333	0	0.955	0.955	0.955	0.571	0.125	0
0	0.587	0.735	0.419	0.301	0.5	0.333	0.5	0.426	0.389	0.615	0.021	0.417	0.333	0.93	0.93	0.93	0.467	0.333	0.333
0	0.604	0.719	0.426	0.375	0.5	0.423	0.558	0.481	0.481	0.66	0.018	0.375	0.333	0.923	0.923	0.923	0.466	0.333	0.333
0	0.709	0.847	0.213	0	0	0.286	0.595	0.405	0.357	0.592	0.033	0.271	0.062	0.97	0.97	0.97	0.548	0.127	0
0	0.516	0.652	0.109	0	0	0.217	0.435	0.261	0.261	0.519	0.035	0.208	0.188	0.932	0.932	0.932	0.324	0.05	0
0	0.534	0.733	0.058	0	0.333	0.238	0.452	0.214	0.214	0.573	0.029	0.196	0	0.925	0.925	0.925	0.367	0.045	0
0	0.434	0.474	0.103	0	0	0.208	0.458	0.292	0.25	0.578	0.021	0.259	0.125	0.93	0.93	0.93	0.278	0.056	0
0	0.489	0.158	0	0	0	0.067	0.483	0.45	0.483	0.448	0.019	0.143	0	0.895	0.895	0.895	0.1	0	0
0	0.534	0.637	0.195	0	0	0.269	0.558	0.404	0.288	0.632	0.039	0.25	0	0.943	0.943	0.943	0.422	0.084	0
0	0.576	0.753	0.229	0.044	0.5	0.217	0.522	0.348	0.261	0.488	0.026	0.15	0.067	0.923	0.923	0.923	0.327	0.067	0
0	0.449	0.768	0.63	0.482	0.375	0.379	0.638	0.638	0.603	0.789	0.029	0.6	0.5	0.961	0.961	0.961	0.667	0.562	0.5
0	0.554	0.703	0.236	0.063	0.333	0.24	0.52	0.44	0.36	0.532	0.031	0.198	0	0.942	0.942	0.942	0.258	0.145	0.056
0	0.537	0.567	0	0	0	0.133	0.467	0.2	0.267	0.519	0.049	0.1	0	0.914	0.914	0.914	0.3	0	0
0	0.531	0.695	0.094	0	0.25	0.194	0.516	0.355	0.355	0.503	0.024	0.188	0	0.934	0.934	0.934	0.312	0.038	0
0	0.556	0.746	0.192	0.08	0	0.267	0.45	0.317	0.283	0.557	0.019	0.302	0.095	0.915	0.915	0.915	0.306	0.074	0.042
0	0.631	0.7	0.546	0.471	0.333	0.474	0.632	0.579	0.579	0.485	0.026	0.389	0.333	0.62	0.62	0.62	0.37	0.333	0.333
0	0.574	0.511	0.082	0	0	0.125	0.521	0.438	0.438	0.68	0.04	0.375	0.125	0.97	0.97	0.97	0.429	0.083	0
0	0.627	0.869	0.534	0.471	0.333	0.417	0.625	0.583	0.583	0.681	0.016	0.625	0.5	0.972	0.972	0.972	0.667	0.5	0.5
0	0.574	0.765	0.175	0.062	0.667	0.364	0.523	0.341	0.295	0.613	0.029	0.402	0	0.941	0.941	0.941	0.317	0.05	0
0	0.573	0.319	0	0	0	0.12	0.52	0.44	0.48	0.511	0.019	0.2	0	0.9	0.9	0.9	0.222	0	0
0	0.606	0.798	0.07	0	0.25	0.286	0.429	0.214	0.179	0.507	0.022	0.231	0.033	0.928	0.928	0.928	0.319	0.03	0
0	0.452	0.481	0.231	0.129	0.5	0.273	0.523	0.341	0.341	0.495	0.017	0.243	0	0.91	0.91	0.91	0.367	0	0
0	0.568	0.624	0.236	0.135	0	0.263	0.474	0.368	0.316	0.547	0.036	0.333	0.125	0.921	0.921	0.921	0.341	0.1	0.056

0	0.571	0.727	0.177	0	0.333	0.28	0.58	0.34	0.3	0.573	0.022	0.196	0.071	0.942	0.942	0.942	0.438	0.094	0
0	0.619	0.813	0.153	0	0.5	0.192	0.404	0.288	0.173	0.384	0.019	0.167	0.042	0.901	0.901	0.901	0.28	0.042	0
0	0.594	0.707	0.212	0.067	0	0.278	0.556	0.278	0.222	0.582	0.033	0.33	0.125	0.946	0.946	0.946	0.436	0.1	0.056
0	0.663	0.793	0.261	0.1	0.833	0.217	0.478	0.348	0.217	0.473	0.018	0.153	0.074	0.922	0.922	0.922	0.319	0.1	0.074
0	0.598	0.882	0.632	0.449	0.5	0.345	0.655	0.621	0.621	0.85	0.021	0.75	0.583	0.971	0.971	0.971	0.714	0.643	0.583
0	0.426	0.583	0.065	0	0	0.158	0.368	0.263	0.263	0.509	0.039	0.171	0	0.896	0.896	0.896	0.238	0.05	0
0	0.666	0.807	0.049	0	0.5	0.238	0.405	0.119	0.119	0.508	0.023	0.194	0	0.902	0.902	0.902	0.324	0.037	0
0	0.407	0.64	0.052	0	0	0.222	0.5	0.352	0.352	0.568	0.028	0.311	0.2	0.939	0.939	0.939	0.425	0.083	0
0	0.425	0.566	0.243	0.065	0.333	0.158	0.395	0.342	0.237	0.569	0.031	0.222	0.167	0.91	0.91	0.91	0.306	0.1	0
0	0.407	0.549	0	0	0	0.148	0.519	0.407	0.444	0.562	0.032	0.182	0	0.962	0.962	0.962	0.4	0	0
0	0.509	0.721	0.19	0	0	0.227	0.523	0.432	0.341	0.512	0.031	0.214	0.071	0.934	0.934	0.934	0.389	0.1	0
0	0.618	0.772	0.523	0.455	0.375	0.357	0.607	0.571	0.571	0.781	0.024	0.643	0.5	0.973	0.973	0.973	0.562	0.5	0.5
0	0.557	0.561	0.094	0.056	0	0.048	0.429	0.238	0.238	0.524	0.033	0.112	0	0.959	0.959	0.959	0.486	0.083	0.045
0	0.603	0.857	0.258	0.129	0.25	0.207	0.466	0.328	0.293	0.509	0.021	0.361	0.278	0.928	0.928	0.928	0.369	0.292	0.25
0	0.648	0.793	0.185	0	0	0.286	0.524	0.381	0.286	0.59	0.026	0.286	0	0.924	0.924	0.924	0.321	0.056	0
0	0.53	0.515	0.064	0	0	0.148	0.315	0.204	0.167	0.45	0.031	0.075	0	0.926	0.926	0.926	0.255	0.028	0
0	0.587	0.818	0.288	0.105	0	0.286	0.536	0.429	0.357	0.582	0.022	0.196	0.071	0.929	0.929	0.929	0.357	0.167	0.056
0	0.589	0.688	0.17	0.041	0.5	0.276	0.5	0.328	0.224	0.543	0.017	0.143	0.083	0.939	0.939	0.939	0.449	0.067	0.037
0	0.584	0.642	0.098	0	0	0.217	0.435	0.304	0.261	0.52	0.03	0.208	0	0.928	0.928	0.928	0.303	0.042	0
0	0.612	0.752	0.202	0	0	0.182	0.455	0.364	0.227	0.576	0.031	0.25	0	0.934	0.934	0.934	0.386	0.05	0
0	0.455	0.304	0.149	0.093	0	0.16	0.52	0.48	0.48	0.435	0.033	0.25	0	0.934	0.934	0.934	0.273	0.182	0.1
0	0.649	0.775	0.273	0.058	0.667	0.28	0.56	0.44	0.32	0.654	0.026	0.227	0.111	0.934	0.934	0.934	0.342	0.042	0
0	0.562	0.806	0.18	0	0	0.214	0.464	0.357	0.25	0.628	0.026	0.229	0.056	0.945	0.945	0.945	0.303	0.088	0
0	0.668	0.827	0.441	0.314	0.25	0.25	0.518	0.411	0.375	0.697	0.02	0.491	0.333	0.958	0.958	0.958	0.426	0.364	0.333
0	0.475	0.57	0.059	0	0.25	0.143	0.482	0.375	0.375	0.465	0.025	0.146	0	0.911	0.911	0.911	0.188	0	0
0	0.546	0.754	0.294	0	0.25	0.25	0.589	0.482	0.375	0.595	0.019	0.268	0.125	0.936	0.936	0.936	0.5	0.17	0
0	0.55	0.646	0.572	0.454	0.333	0.421	0.605	0.605	0.553	0.724	0.031	0.571	0.5	0.955	0.955	0.955	0.556	0.5	0.5
0	0.631	0.855	0.218	0.115	0.333	0.273	0.477	0.341	0.295	0.556	0.038	0.31	0.056	0.95	0.95	0.95	0.412	0.1	0.056
0	0.654	0.796	0.218	0.152	0.25	0.3	0.567	0.433	0.4	0.708	0.019	0.482	0.357	0.959	0.959	0.959	0.553	0.333	0.3
0	0.548	0.733	0.2	0.053	0	0.095	0.381	0.238	0.143	0.475	0.022	0.22	0	0.911	0.911	0.911	0.235	0.056	0.03
0	0.545	0.565	0	0	0	0.176	0.382	0.088	0.147	0.537	0.033	0.125	0	0.934	0.934	0.934	0.286	0	0
0	0.531	0.71	0	0	0	0.182	0.341	0.159	0.205	0.337	0.032	0.089	0	0.871	0.871	0.871	0.133	0	0
0	0.54	0.662	0.085	0	0	0.148	0.537	0.463	0.463	0.611	0.024	0.167	0	0.941	0.941	0.941	0.375	0.143	0
0	0.551	0.628	0.174	0	0	0.143	0.571	0.476	0.429	0.746	0.032	0.286	0.286	0.964	0.964	0.964	0.667	0.222	0
0	0.586	0.754	0.115	0	0	0.211	0.447	0.184	0.184	0.634	0.039	0.196	0	0.94	0.94	0.94	0.333	0	0
0	0.47	0.759	0.147	0.054	0.25	0.138	0.466	0.397	0.362	0.526	0.022	0.214	0	0.929	0.929	0.929	0.311	0.045	0
0	0.55	0.367	0	0	0	0.167	0.458	0.25	0.292	0.52	0.036	0.134	0	0.912	0.912	0.912	0.211	0	0
0	0.529	0.718	0.226	0.066	0.5	0.174	0.478	0.435	0.348	0.479	0.031	0.25	0.188	0.915	0.915	0.915	0.25	0.111	0.062
0	0.723	0.891	0.601	0.501	0.75	0.706	0.765	0.647	0.588	0.799	0.025	0.688	0.5	0.978	0.978	0.978	0.722	0.545	0.5
0	0.608	0.792	0.218	0	0.333	0.217	0.391	0.261	0.087	0.554	0.036	0.179	0	0.935	0.935	0.935	0.29	0.092	0
0	0.662	0.784	0.25	0.056	0.333	0.364	0.591	0.364	0.227	0.542	0.029	0.302	0.159	0.955	0.955	0.955	0.454	0.151	0.037
0	0.51	0.774	0.171	0	0.333	0.286	0.524	0.333	0.238	0.569	0.029	0.268	0.071	0.951	0.951	0.951	0.354	0.056	0
0	0.412	0.267	0	0	0	0.071	0.482	0.446	0.482	0.483	0.025	0.2	0	0.9	0.9	0.9	0.222	0	0
0	0.438	0.471	0.169	0	0	0.19	0.5	0.452	0.405	0.512	0.04	0.25	0	0.946	0.946	0.946	0.333	0.1	0
0	0.509	0.488	0.146	0	0	0.179	0.518	0.411	0.339	0.514	0.03	0.167	0	0.907	0.907	0.907	0.293	0.056	0
0	0.675	0.805	0.497	0.453	0.333	0.5	0.688	0.604	0.604	0.821	0.035	0.714	0.5	0.974	0.974	0.974	0.625	0.5	0.5
0	0.481	0.711	0.136	0	0	0.179	0.482	0.339	0.304	0.468	0.026	0.174	0	0.955	0.955	0.955	0.306	0.042	0
0	0.54	0.658	0.253	0.111	0.667	0.261	0.457	0.326	0.283	0.561	0.031	0.268	0	0.945	0.945	0.945	0.359	0.045	0
0	0.534	0.761	0.173	0	0.333	0.238	0.405	0.31	0.214	0.53	0.027	0.205	0	0.941	0.941	0.941	0.341	0.1	0
0	0.545	0.657	0.164	0	0	0.25	0.429	0.25	0.179	0.513	0.032	0.186	0.061	0.942	0.942	0.942	0.369	0.083	0
0	0.53	0.641	0.104	0	0.667	0.15	0.375	0.225	0.175	0.549	0.035	0.214	0.111	0.951	0.951	0.951	0.438	0.038	0
0	0.577	0.712	0.19	0	0	0.2	0.46	0.3	0.22	0.639	0.029	0.299	0.188	0.966	0.966	0.966	0.562	0.095	0
0	0.597	0.739	0.534	0.492	0.333	0.417	0.625	0.625	0.625	0.764	0.029	0.625	0.5	0.96	0.96	0.96	0.625	0.545	0.5
0	0.521	0.806	0.566	0.472	0.333	0.5	0.75	0.667	0.625	0.798	0.026	0.714	0.643	0.967	0.967	0.967	0.714	0.562	0.5
0	0.43	0.586	0.178	0.052	0.375	0.194	0.484	0.387	0.355	0.518	0.02	0.214	0	0.928	0.928	0.928	0.336	0.038	0
0	0.627	0.714	0.083	0	0.25	0.143	0.536	0.464	0.464	0.694	0.033	0.286	0	0.944	0.944	0.944	0.5	0.1	0
0	0.592	0.791	0.132	0.05	0	0.143	0.429	0.19	0.143	0.48	0.026	0.236	0.056	0.92	0.92	0.92	0.342	0.056	0.03
0	0.453	0.526	0.085	0	0	0.094	0.516	0.484	0.484	0.633	0.03	0.286	0.1	0.963	0.963	0.963	0.333	0.083	0
0	0.463	0.436	0.101	0	0	0.24	0.48	0.32	0.28	0.522	0.039	0.325	0	0.926	0.926	0.926	0.317	0.05	0
0	0.658	0.791	0.051	0	0.333	0.1	0.425	0.125	0.125	0.509	0.021	0.145	0	0.917	0.917	0.917	0.291	0	0
0	0.495	0.613	0.052	0	0.333	0.192	0.519	0.327	0.327	0.529	0.027	0.183	0	0.934	0.934	0.934	0.35	0	0
0	0.324	0.36	0	0	0	0.107	0.518	0.446	0.482	0.55	0.032	0.25	0	0.933	0.933	0.933	0.2	0	0

0	0.59	0.697	0.264	0	0	0.276	0.448	0.379	0.172	0.508	0.025	0.255	0	0.926	0.926	0.926	0.341	0.07	0
0	0.567	0.719	0.049	0	0.5	0.136	0.386	0.114	0.114	0.479	0.028	0.159	0	0.928	0.928	0.928	0.257	0	0
0	0.439	0.414	0	0	0	0.167	0.312	0.146	0.188	0.335	0.033	0.095	0	0.879	0.879	0.879	0.141	0	0
0	0.611	0.855	0.457	0.365	0.25	0.375	0.609	0.516	0.516	0.792	0.027	0.625	0.5	0.977	0.977	0.977	0.65	0.5	0.5
0	0.451	0.526	0.057	0	0.333	0.167	0.361	0.139	0.139	0.323	0.033	0.125	0	0.619	0.619	0.619	0.15	0	0
0	0.511	0.524	0	0	0.333	0.067	0.267	0.067	0.133	0.492	0.039	0.1	0	0.889	0.889	0.889	0.167	0	0
0	0.493	0.685	0.137	0	0	0.24	0.5	0.3	0.26	0.559	0.027	0.199	0	0.958	0.958	0.958	0.354	0.1	0
0	0.515	0.457	0.099	0	0	0.222	0.528	0.417	0.417	0.619	0.032	0.286	0	0.952	0.952	0.952	0.5	0.091	0
0	0.479	0.471	0	0	0	0.095	0.405	0.31	0.357	0.242	0.036	0.1	0	0.884	0.884	0.884	0.127	0	0
0	0.493	0.621	0.218	0.058	0.25	0.16	0.44	0.4	0.32	0.562	0.027	0.286	0.071	0.918	0.918	0.918	0.312	0.056	0
0	0.564	0.594	0.049	0	0	0.208	0.458	0.25	0.25	0.52	0.035	0.062	0	0.925	0.925	0.925	0.2	0	0
0	0.501	0.591	0	0	0	0.179	0.482	0.304	0.339	0.548	0.032	0.226	0	0.931	0.931	0.931	0.306	0	0
0	0.615	0.669	0.106	0.063	0.333	0.25	0.45	0.25	0.25	0.582	0.031	0.268	0	0.934	0.934	0.934	0.261	0	0
0	0.575	0.652	0.203	0	0.333	0.2	0.46	0.38	0.3	0.512	0.038	0.254	0.056	0.946	0.946	0.946	0.278	0.045	0
0	0.582	0.708	0.18	0	0	0.308	0.538	0.308	0.269	0.642	0.038	0.25	0.071	0.947	0.947	0.947	0.45	0.045	0
0	0.4	0.376	0.097	0	0	0.147	0.471	0.382	0.382	0.218	0.026	0.143	0	0.916	0.916	0.916	0.153	0.045	0
0	0.55	0.585	0.135	0	0.25	0.192	0.481	0.327	0.288	0.628	0.029	0.208	0	0.945	0.945	0.945	0.45	0.045	0
0	0.563	0.734	0.51	0.441	0.5	0.476	0.643	0.595	0.595	0.733	0.031	0.583	0.5	0.949	0.949	0.949	0.562	0.5	0.5
0	0.393	0.361	0.081	0	0	0.105	0.447	0.395	0.395	0.55	0.04	0.25	0	0.928	0.928	0.928	0.2	0.1	0
0	0.609	0.856	0.189	0	0	0.222	0.481	0.37	0.296	0.497	0.029	0.234	0.091	0.979	0.979	0.979	0.598	0.111	0
0	0.4	0.293	0.164	0	0.5	0.1	0.325	0.325	0.225	0.321	0.031	0.214	0	0.906	0.906	0.906	0.25	0.05	0
0	0.571	0.644	0.105	0.06	0.333	0.28	0.5	0.34	0.34	0.582	0.022	0.259	0	0.94	0.94	0.94	0.347	0.045	0
0	0.532	0.714	0.183	0	0	0.172	0.466	0.362	0.293	0.585	0.027	0.268	0	0.948	0.948	0.948	0.422	0.087	0
0	0.621	0.807	0.152	0	0	0.28	0.52	0.32	0.28	0.56	0.029	0.278	0	0.953	0.953	0.953	0.536	0.045	0
0	0.645	0.802	0.38	0.049	0.25	0.357	0.607	0.464	0.286	0.568	0.024	0.204	0.089	0.93	0.93	0.93	0.444	0.162	0.042
0	0.407	0.586	0	0	0	0.185	0.5	0.315	0.352	0.512	0.024	0.143	0	0.905	0.905	0.905	0.225	0	0
0	0.522	0.617	0.126	0	0.333	0.176	0.412	0.235	0.176	0.495	0.031	0.271	0.083	0.93	0.93	0.93	0.35	0.042	0
0	0.613	0.848	0.298	0.047	0	0.276	0.483	0.379	0.276	0.573	0.027	0.278	0.042	0.936	0.936	0.936	0.351	0.074	0.037
0	0.598	0.773	0.12	0	0	0.263	0.474	0.263	0.211	0.507	0.039	0.208	0	0.936	0.936	0.936	0.354	0.056	0
0	0.571	0.901	0.293	0.057	0	0.267	0.55	0.45	0.383	0.676	0.024	0.443	0.1	0.964	0.964	0.964	0.536	0.117	0
0	0.567	0.77	0.138	0	0.333	0.3	0.525	0.375	0.325	0.546	0.031	0.325	0.062	0.918	0.918	0.918	0.317	0.05	0
0	0.342	0.393	0	0	0	0.065	0.452	0.419	0.452	0.524	0.033	0.143	0	0.922	0.922	0.922	0.333	0	0
0	0.522	0.273	0	0	0	0.1	0.533	0.467	0.5	0.417	0.019	0.25	0	0.941	0.941	0.941	0.333	0	0
0	0.46	0.465	0.051	0	0	0.154	0.442	0.288	0.288	0.54	0.032	0.194	0	0.917	0.917	0.917	0.155	0	0
0	0.438	0.596	0.08	0	0	0.217	0.413	0.239	0.239	0.515	0.043	0.196	0	0.934	0.934	0.934	0.286	0.038	0
0	0.598	0.659	0.228	0	0.5	0.273	0.455	0.273	0.182	0.562	0.031	0.268	0.048	0.918	0.918	0.918	0.231	0.037	0
0	0.661	0.839	0.568	0.431	0.5	0.458	0.667	0.625	0.583	0.781	0.024	0.643	0.5	0.953	0.953	0.953	0.6	0.556	0.5
0	0.745	0.763	0.573	0.39	0.667	0.565	0.674	0.543	0.457	0.739	0.02	0.524	0.381	0.94	0.94	0.94	0.55	0.37	0.333
0	0.468	0.489	0.048	0	0.25	0.192	0.442	0.288	0.288	0.526	0.03	0.143	0	0.919	0.919	0.919	0.325	0	0
0	0.579	0.853	0.254	0.052	0.25	0.233	0.55	0.417	0.35	0.667	0.024	0.292	0	0.965	0.965	0.965	0.422	0.045	0
0	0.524	0.63	0.052	0	0	0.179	0.464	0.357	0.357	0.587	0.022	0.321	0	0.921	0.921	0.921	0.333	0.1	0
0	0.578	0.808	0.144	0	0	0.222	0.37	0.259	0.148	0.517	0.025	0.194	0	0.929	0.929	0.929	0.306	0.075	0
0	0.591	0.592	0.171	0	0.25	0.103	0.517	0.517	0.483	0.562	0.02	0.286	0.143	0.942	0.942	0.942	0.5	0.167	0
0	0.505	0.385	0.17	0.063	0.5	0.238	0.452	0.31	0.31	0.582	0.024	0.259	0	0.921	0.921	0.921	0.333	0.045	0
0	0.37	0.265	0	0	0	0.111	0.5	0.444	0.5	0.5	0.025	0.25	0	0.919	0.919	0.919	0.2	0	0
0	0.547	0.669	0.055	0	0	0.286	0.452	0.262	0.262	0.48	0.033	0.211	0	0.93	0.93	0.93	0.279	0.038	0
0	0.417	0.633	0.258	0.057	0	0.214	0.464	0.429	0.357	0.498	0.021	0.225	0.071	0.931	0.931	0.931	0.295	0.056	0
0	0.596	0.636	0	0	0	0.194	0.516	0.29	0.323	0.561	0.022	0.208	0	0.934	0.934	0.934	0.275	0	0
0	0.492	0.667	0.149	0	0.25	0.321	0.554	0.375	0.304	0.581	0.019	0.292	0	0.917	0.917	0.917	0.261	0.101	0
0	0.497	0.658	0.055	0	0	0.19	0.452	0.214	0.214	0.565	0.033	0.268	0	0.95	0.95	0.95	0.449	0	0
0	0.646	0.58	0.195	0	0	0.312	0.562	0.438	0.375	0.69	0.047	0.375	0.125	0.954	0.954	0.954	0.444	0.182	0
0	0.634	0.799	0.491	0.358	0.25	0.444	0.574	0.5	0.463	0.506	0.019	0.336	0.25	0.703	0.703	0.703	0.356	0.25	0.25
0	0.436	0.624	0.105	0	0	0.105	0.263	0.105	0.105	0.348	0.02	0.103	0	0.856	0.856	0.856	0.125	0	0
0	0.491	0.527	0	0	0	0.182	0.409	0.273	0.318	0.446	0.031	0.167	0	0.901	0.901	0.901	0.312	0	0
0	0.55	0.667	0.186	0	0.333	0.19	0.452	0.31	0.214	0.596	0.027	0.268	0	0.934	0.934	0.934	0.35	0.1	0
0	0.474	0.303	0	0	0	0.077	0.308	0.154	0.231	0.232	0.041	0.071	0	0.874	0.874	0.874	0.188	0	0
0	0.802	0.827	0.641	0.609	0.5	0.714	0.75	0.679	0.679	0.742	0.016	0.6	0.5	0.938	0.938	0.938	0.667	0.5	0.5
0	0.357	0.291	0	0	0	0.107	0.536	0.464	0.5	0.583	0.033	0.125	0	0.919	0.919	0.919	0.222	0	0
0	0.64	0.752	0.15	0.056	0	0.238	0.476	0.19	0.19	0.428	0.025	0.25	0	0.891	0.891	0.891	0.268	0.042	0
0	0.459	0.314	0	0	0	0.111	0.519	0.444	0.481	0.644	0.032	0.222	0.111	0.934	0.934	0.934	0.3	0	0
0	0.405	0.636	0.115	0	0	0.161	0.516	0.419	0.387	0.51	0.028	0.175	0.062	0.938	0.938	0.938	0.261	0.112	0
0	0.529	0.806	0.177	0	0.375	0.258	0.5	0.403	0.306	0.565	0.025	0.268	0.062	0.945	0.945	0.945	0.367	0.045	0

0	0.637	0.829	0.539	0.404	0.25	0.357	0.607	0.571	0.571	0.762	0.025	0.571	0.5	0.95	0.95	0.95	0.611	0.5	0.5
0	0.545	0.718	0	0	0.25	0.258	0.516	0.29	0.323	0.582	0.022	0.311	0	0.931	0.931	0.931	0.307	0	0
0	0.329	0.276	0	0	0	0.13	0.522	0.435	0.478	0.448	0.025	0.167	0	0.908	0.908	0.908	0.286	0	0
0	0.54	0.751	0.275	0	0	0.233	0.583	0.45	0.383	0.524	0.024	0.268	0.062	0.928	0.928	0.928	0.393	0.106	0
0	0.441	0.325	0	0	0	0.167	0.292	0.125	0.208	0.278	0.033	0.083	0	0.853	0.853	0.853	0.188	0	0
0	0.478	0.516	0.109	0	0	0.2	0.375	0.225	0.175	0.526	0.043	0.292	0.056	0.927	0.927	0.927	0.306	0.036	0
0	0.481	0.635	0	0	0	0.19	0.357	0.167	0.214	0.486	0.033	0.188	0	0.918	0.918	0.918	0.2	0	0
0	0.451	0.566	0.107	0	0	0.143	0.381	0.19	0.19	0.531	0.039	0.188	0	0.934	0.934	0.934	0.261	0	0
0	0.627	0.818	0.215	0.062	0.5	0.375	0.583	0.417	0.333	0.616	0.021	0.357	0.071	0.942	0.942	0.942	0.389	0.145	0.056
0	0.487	0.45	0.059	0	0.333	0.222	0.389	0.222	0.222	0.508	0.026	0.211	0.056	0.901	0.901	0.901	0.238	0	0
0	0.467	0.48	0	0	0	0.143	0.411	0.304	0.339	0.496	0.027	0.171	0	0.905	0.905	0.905	0.31	0	0
0	0.559	0.437	0.16	0	0	0.161	0.548	0.452	0.452	0.562	0.033	0.286	0	0.947	0.947	0.947	0.429	0.125	0
0	0.406	0.574	0.243	0.076	0	0.185	0.519	0.481	0.444	0.57	0.025	0.333	0	0.971	0.971	0.971	0.3	0.182	0.1
0	0.301	0.553	0.079	0	0	0.138	0.534	0.466	0.466	0.556	0.053	0.333	0	0.958	0.958	0.958	0.429	0.111	0
0	0.632	0.764	0.103	0	0	0.304	0.543	0.239	0.239	0.602	0.031	0.196	0	0.943	0.943	0.943	0.359	0.042	0
0	0.47	0.676	0.094	0	0	0.095	0.524	0.476	0.476	0.589	0.04	0.167	0	0.94	0.94	0.94	0.5	0.143	0
0	0.683	0.831	0.541	0.398	0.625	0.4	0.617	0.55	0.483	0.778	0.019	0.597	0.5	0.964	0.964	0.964	0.656	0.5	0.467
0	0.643	0.564	0	0	0	0.2	0.56	0.4	0.44	0.743	0.033	0.429	0	0.968	0.968	0.968	0.714	0	0
0	0.552	0.673	0.045	0	0	0.2	0.4	0.2	0.2	0.353	0.035	0.13	0	0.909	0.909	0.909	0.217	0.028	0
0	0.57	0.759	0.037	0	0.25	0.222	0.426	0.204	0.204	0.537	0.015	0.258	0	0.909	0.909	0.909	0.294	0.037	0
0	0.542	0.84	0.297	0.115	0	0.25	0.536	0.464	0.429	0.513	0.03	0.324	0.125	0.958	0.958	0.958	0.299	0.17	0.071
0	0.509	0.651	0	0	0	0.111	0.537	0.463	0.5	0.611	0.033	0.333	0	0.963	0.963	0.963	0.5	0	0
0	0.516	0.628	0.115	0.061	0.333	0.286	0.452	0.262	0.262	0.365	0.016	0.181	0	0.883	0.883	0.883	0.223	0.03	0
0	0.517	0.654	0.271	0.105	0	0.267	0.517	0.417	0.383	0.613	0.028	0.411	0.071	0.942	0.942	0.942	0.412	0.142	0.056
0	0.504	0.54	0.065	0	0	0.263	0.447	0.237	0.237	0.591	0.039	0.214	0	0.928	0.928	0.928	0.211	0.05	0
0	0.43	0.554	0.222	0	0	0.2	0.433	0.3	0.167	0.275	0.041	0.181	0.062	0.934	0.934	0.934	0.267	0.095	0
0	0.499	0.657	0.169	0	0	0.227	0.455	0.318	0.273	0.525	0.039	0.288	0.062	0.909	0.909	0.909	0.306	0.05	0
0	0.56	0.666	0.155	0.057	0.25	0.25	0.536	0.393	0.357	0.676	0.027	0.312	0.167	0.952	0.952	0.952	0.5	0.212	0.1
0	0.533	0.667	0.087	0	0.5	0.192	0.462	0.308	0.308	0.52	0.024	0.125	0	0.932	0.932	0.932	0.286	0	0
0	0.541	0.685	0.129	0.096	0	0.207	0.5	0.328	0.328	0.675	0.024	0.417	0.062	0.965	0.965	0.965	0.562	0.15	0.111
0	0.465	0.524	0.044	0	0	0.208	0.417	0.25	0.25	0.554	0.046	0.196	0	0.934	0.934	0.934	0.247	0.038	0
0	0.549	0.641	0.073	0	0	0.161	0.468	0.274	0.274	0.533	0.032	0.15	0.05	0.935	0.935	0.935	0.417	0.036	0
0	0.441	0.487	0	0	0	0.067	0.267	0.133	0.2	0.298	0.033	0.143	0	0.874	0.874	0.874	0.2	0	0
0	0.492	0.603	0	0	0	0.158	0.474	0.368	0.421	0.54	0.032	0.143	0	0.929	0.929	0.929	0.429	0	0
0	0.532	0.745	0.471	0.374	0.375	0.37	0.611	0.574	0.537	0.781	0.028	0.571	0.5	0.953	0.953	0.953	0.643	0.5	0.5
0	0.55	0.703	0.052	0	0	0.217	0.435	0.261	0.261	0.525	0.036	0.111	0	0.933	0.933	0.933	0.411	0.036	0
0	0.516	0.507	0	0	0	0.107	0.464	0.393	0.429	0.49	0.03	0.125	0	0.949	0.949	0.949	0.4	0	0
0	0.6	0.815	0.125	0.051	0	0.276	0.603	0.362	0.328	0.583	0.026	0.325	0.056	0.957	0.957	0.957	0.488	0.127	0.042
0	0.456	0.571	0	0	0	0.125	0.406	0.219	0.281	0.27	0.058	0.071	0	0.919	0.919	0.919	0.157	0	0
0	0.58	0.811	0.365	0.132	0.25	0.242	0.53	0.47	0.409	0.637	0.021	0.375	0.071	0.951	0.951	0.951	0.312	0.188	0.111
0	0.546	0.583	0.264	0.058	0.375	0.259	0.556	0.444	0.37	0.495	0.019	0.243	0.071	0.91	0.91	0.91	0.31	0.1	0.056
0	0.586	0.707	0.201	0.031	0.375	0.2	0.4	0.267	0.2	0.577	0.024	0.222	0	0.935	0.935	0.935	0.324	0.053	0
0	0.56	0.746	0.225	0	0.5	0.227	0.5	0.364	0.227	0.531	0.029	0.196	0.062	0.944	0.944	0.944	0.388	0.05	0
0	0.62	0.719	0.076	0	0	0.259	0.5	0.241	0.204	0.537	0.04	0.195	0.048	0.925	0.925	0.925	0.283	0.028	0
0	0.503	0.676	0.047	0	0.5	0.13	0.413	0.239	0.239	0.507	0.029	0.174	0.062	0.937	0.937	0.937	0.325	0	0
0	0.65	0.783	0.041	0	0.375	0.233	0.483	0.317	0.317	0.613	0.024	0.25	0.188	0.938	0.938	0.938	0.275	0.036	0
0	0.444	0.518	0.071	0	0	0.238	0.476	0.333	0.333	0.46	0.029	0.155	0	0.904	0.904	0.904	0.167	0.05	0
0	0.439	0.664	0.117	0	0	0.15	0.375	0.275	0.225	0.294	0.035	0.111	0	0.934	0.934	0.934	0.222	0.056	0
0	0.377	0.583	0.091	0	0	0.13	0.5	0.457	0.457	0.54	0.044	0.286	0	0.927	0.927	0.927	0.333	0.1	0
0	0.553	0.718	0.211	0.117	0.333	0.217	0.413	0.37	0.326	0.515	0.029	0.188	0	0.929	0.929	0.929	0.267	0.091	0.05
0	0.602	0.778	0.462	0.101	0	0.25	0.5	0.458	0.25	0.504	0.036	0.236	0.042	0.924	0.924	0.924	0.333	0.097	0.042
0	0.528	0.651	0	0	0	0.182	0.409	0.182	0.227	0.493	0.029	0.117	0	0.922	0.922	0.922	0.306	0	0
0	0.542	0.672	0.063	0	0	0.167	0.5	0.333	0.333	0.49	0.036	0.143	0	0.922	0.922	0.922	0.267	0.045	0
0	0.539	0.712	0.21	0.13	0.375	0.231	0.442	0.25	0.25	0.569	0.019	0.276	0.048	0.918	0.918	0.918	0.288	0.037	0
0	0.579	0.841	0.487	0.356	0.375	0.233	0.567	0.533	0.5	0.55	0.019	0.389	0.333	0.944	0.944	0.944	0.524	0.333	0.333
0	0.559	0.656	0.118	0.05	0.333	0.154	0.442	0.25	0.25	0.469	0.034	0.122	0	0.92	0.92	0.92	0.258	0	0
0	0.502	0.689	0.197	0	0.333	0.25	0.5	0.4	0.3	0.543	0.026	0.286	0	0.93	0.93	0.93	0.375	0.111	0
0	0.452	0.244	0	0	0	0.036	0.446	0.411	0.446	0.54	0.04	0.091	0	0.927	0.927	0.927	0.222	0	0
0	0.534	0.666	0.147	0.061	0	0.217	0.435	0.304	0.304	0.577	0.031	0.339	0	0.945	0.945	0.945	0.35	0.125	0.062
0	0.478	0.609	0.107	0	0	0.15	0.425	0.275	0.275	0.458	0.026	0.083	0	0.897	0.897	0.897	0.253	0.045	0
0	0.523	0.608	0.24	0.057	0	0.267	0.55	0.45	0.383	0.536	0.024	0.267	0.056	0.933	0.933	0.933	0.259	0.091	0.045
0	0.348	0.341	0	0	0	0.105	0.395	0.237	0.289	0.232	0.026	0.127	0	0.902	0.902	0.902	0.193	0	0

0	0.488	0.593	0	0	0	0.19	0.5	0.357	0.405	0.524	0.032	0.125	0	0.941	0.941	0.941	0.625	0	0
0	0.499	0.435	0.059	0	0	0.19	0.429	0.286	0.286	0.559	0.033	0.238	0	0.901	0.901	0.901	0.222	0	0
0	0.649	0.771	0.173	0	0	0.316	0.447	0.237	0.184	0.545	0.033	0.196	0.062	0.938	0.938	0.938	0.333	0.056	0
0	0.536	0.524	0.093	0	0	0.214	0.5	0.357	0.321	0.503	0.024	0.155	0	0.92	0.92	0.92	0.343	0.042	0
0	0.451	0.538	0.056	0	0	0.231	0.5	0.346	0.346	0.539	0.036	0.127	0	0.924	0.924	0.924	0.211	0.045	0
0	0.646	0.817	0.504	0.472	0.5	0.409	0.659	0.614	0.614	0.742	0.031	0.5	0.5	0.945	0.945	0.945	0.643	0.5	0.5
0	0.498	0.648	0.049	0	0	0.222	0.5	0.315	0.315	0.526	0.035	0.226	0	0.915	0.915	0.915	0.236	0.045	0
0	0.505	0.687	0.118	0.069	0	0.261	0.543	0.37	0.37	0.561	0.026	0.238	0.167	0.937	0.937	0.937	0.35	0.071	0
0	0.447	0.577	0	0	0.333	0.217	0.435	0.217	0.261	0.581	0.042	0.205	0	0.926	0.926	0.926	0.341	0	0
0	0.619	0.849	0.227	0.089	0.333	0.16	0.34	0.22	0.18	0.456	0.031	0.22	0	0.95	0.95	0.95	0.324	0.028	0
0	0.641	0.707	0.254	0.094	0.333	0.19	0.548	0.5	0.452	0.601	0.032	0.375	0	0.967	0.967	0.967	0.5	0.182	0.1
0	0.536	0.56	0	0	0	0.238	0.595	0.405	0.452	0.556	0.032	0.25	0	0.947	0.947	0.947	0.4	0	0
0	0.54	0.682	0.055	0	0.333	0.227	0.432	0.25	0.25	0.565	0.031	0.205	0	0.943	0.943	0.943	0.303	0	0
0	0.406	0.468	0	0	0	0.103	0.431	0.293	0.328	0.319	0.021	0.125	0	0.911	0.911	0.911	0.278	0	0
0	0.522	0.578	0.298	0.186	0.333	0.318	0.523	0.432	0.386	0.613	0.022	0.286	0.071	0.943	0.943	0.943	0.45	0.167	0.056
0	0.335	0.263	0	0	0	0.105	0.474	0.421	0.474	0	0.032	0	0	0.892	0.892	0.892	0.286	0	0
0	0.5	0.713	0.407	0.314	0.375	0.333	0.583	0.517	0.517	0.781	0.019	0.571	0.5	0.95	0.95	0.95	0.611	0.5	0.5
0	0.612	0.917	0.534	0.412	0.333	0.375	0.667	0.583	0.542	0.764	0.017	0.562	0.5	0.973	0.973	0.973	0.688	0.545	0.5
0	0.492	0.658	0.059	0	0	0.231	0.519	0.365	0.365	0.558	0.026	0.236	0	0.935	0.935	0.935	0.364	0	0
0	0.481	0.514	0.08	0	0.5	0.071	0.321	0.179	0.179	0.264	0.041	0.062	0	0.898	0.898	0.898	0.199	0	0
0	0.564	0.751	0.171	0	0.333	0.286	0.476	0.333	0.238	0.571	0.031	0.205	0	0.943	0.943	0.943	0.386	0.062	0
0	0.552	0.773	0.073	0	0.25	0.233	0.483	0.283	0.25	0.363	0.019	0.192	0	0.931	0.931	0.931	0.329	0.037	0
0	0.508	0.639	0.052	0	0	0.185	0.5	0.352	0.352	0.551	0.039	0.125	0.062	0.934	0.934	0.934	0.402	0.036	0
0	0.551	0.771	0.214	0	0	0.152	0.455	0.394	0.303	0.565	0.025	0.222	0	0.951	0.951	0.951	0.486	0.056	0
0	0.624	0.517	0.183	0	0	0.138	0.552	0.483	0.483	0.7	0.024	0.333	0.167	0.952	0.952	0.952	0.429	0.125	0
0	0.647	0.882	0.3	0	0.25	0.345	0.569	0.466	0.259	0.646	0.021	0.347	0.056	0.953	0.953	0.953	0.464	0.088	0
0	0.479	0.633	0.155	0	0	0.208	0.458	0.333	0.25	0.533	0.032	0.196	0	0.938	0.938	0.938	0.27	0.062	0
0	0.43	0.293	0	0	0.5	0.091	0.364	0.364	0.455	0.467	0.047	0.2	0	0.868	0.868	0.868	0.125	0	0
0	0.471	0.677	0.139	0.075	0	0.235	0.441	0.265	0.265	0.525	0.031	0.298	0.071	0.931	0.931	0.931	0.229	0.125	0.071
0	0.458	0.466	0	0	0	0.107	0.518	0.446	0.482	0.541	0.04	0.222	0	0.94	0.94	0.94	0.375	0	0

Statement of authorship

I hereby certify that this bachelor/master thesis has been composed by myself and describes my own work, unless otherwise acknowledged in the test. All references and verbatim extracts hove been quoted and all sources of information have been specifically acknowledged. It has not been accepted in any previous application for a degree.

Leipzig, 26 August 2021

Enzo Lima