LPD - Computing Assignment

Najwa Laabid, University of Eastern Finland May 22, 2020

Contents

1 Tasks 1 & 2 **3**
1.1 Methodology . . . 3
1.2 Datasets . . . 3
1.2.1 *House* dataset - congressional votes . . . 3
1.2.2 *Bats* dataset - bats' species across Europe . . . 4
1.2.3 *Retail* - basket data . . . 5
1.2.4 *Groceries* - shopping data . . . 5
1.2.5 *Adult* - income prediction . . . 5
1.2.6 *Accidents* - traffic accidents . . . 6
1.2.7 *Abalone* - sea snails' data . . . 6
1.3 Results . . . 6
1.4 Conclusion . . . 7

2 Task 3 **8**
2.1 *Apriori* algorithm . . . 8
2.2 Generating association rules . . . 8

3 Task 4 **9**
3.1 IBM Generator Data . . . 9
3.2 Synthetic data . . . 9

A General statistics **10**

B Binary Matrices **10**

C Items' distribution **17**

D Transactions' length distribution **17**

E Mining closed itemsets **18**
E.1 Running times . . . 18
E.2 Statistics . . . 19

F Mining maximal itemsets **21**
F.1 Running times . . . 21
F.2 Statistics . . . 22

G Mining association rules **23**
G.1 Running times 23
G.2 Statistics 24

1 Tasks 1 & 2

Tasks 1 and 2 are about comparing the performance of two algorithms, `fp-growth` and `eclat`, in mining maximal and closed itemsets and association rules. The comparison between the algorithms will cover their outputs and execution time. Before performing this comparison however, we need to prepare the datasets to fit the input format expected by each algorithm, and compute exploratory statistics of the data to justify the algorithms' performance and output. This report is organized as follows: the methodology section gives an overview of the steps taken in the analysis, results reports on the characteristics and running results for every dataset, and conclusion provides a summary of the data characteristics observed to have an effect on the mining algorithms.

1.1 Methodology

This section provides an overview of the content of section 1 (Tasks 1 & 2). We first start by describing the domain of each dataset and the circumstances of data collection. We then explain the format of the given data and how it can be formatted to match the algorithms' expectation as implemented by Christian Borgelt (see [1]). Both descriptions can be found in section Datasets under the subsection corresponding to each dataset.

Section Results is dedicated to describing the main statistical characteristics differentiating the datasets, and reporting on the algorithms' performance throughout. We evaluated the algorithms at different support levels for each dataset, on the tasks of mining closed and maximal itemsets, and association rules. Each dataset was mined within the support interval best showing changes in its results.

As mentioned before, the evaluation targeted the running time as well as output characteristics. To compare run times, we plotted runtime over support values for each dataset. To evaluate the mining results (itemsets and rules), we report the number of mined sets, and the min, max, average, and variance of the size of the sets. All plots can be found in the appendices A to F. The results and/or plots were generated by functions `analyze_mining_results` (`assignment_code.py`, l:111), based on the outputs given by function `get_mining_results`. A description of the work of both functions can be found in the comments of the code file `assignment_code.py`.

1.2 Datasets

1.2.1 *House* dataset - congressional votes

Description This dataset was collected from the votes of congressmen on 16 key votes during the year 1984. The data comes in the form of a multi-line file, each line representing votes of one congressman. The first attribute is a Boolean 'class' variable, which takes values 'republican' (0) or 'democrat' (1) depending on the political affiliation of the voter. Each of the categorical attributes thereafter maps to one of the 16 topics of vote, and accepts three possible values: yes (y), no (n), and unknown (?). The dataset itself is in file `house.data`. Information on

the attributes' names and encoding is in the file `house.bininfo`, while the file `house.names` contains a general description of the data, the context of its collection, and the studies that have used it in the past. Analyzing this dataset can reveal patterns in the republican/democrats positions, which can serve as basis for predicting the outcome of future voting sessions and/or general political developments.

Loading data To load the data of *House*, we use the content of the files `house.bininfo` and `house.data` as follows:

First, we get information on the attributes' type and values from `house.bininfo`. Each attribute gives a tuple of three variables: name (the name of the variable), values (possible values for the attribute), type (boolean, categorical, or positional), and offset (`offset[i]=offset[i-1]+len(values[i-1])`). The offset is used to give every categorical variable a unique range of values. To illustrate this transformation, let us consider attribute 4 (physician-fee-freeze). The attributes preceding it have a total of 11 values between them. We therefore expect attribute 4 to have an offset of 11, while each of the three values it can take maps to values 11, 12 and 13 in turn. The code carrying out this step can be found in function `read_bininfo` (file:`helper.py`, l: 182).

Second, we use the offset and values of each attribute to encode transactions numerically. The output of this step is equivalent to the original database, except that the value of each attribute has been converted to offset + index of value in the list of values. This step serves to prepare the dataset for mining by transforming every record to a set of unique items. The implementation is in function `load_data_txt` (file:`assignment_code.py`, l: 155). The output of this step can be saved in a file to be used by the algorithms. The file in this case is `X_house.dat`.

1.2.2 *Bats* dataset - bats' species across Europe

Description This dataset contains information on the bats' species across Europe. The data is generated by placing a grid with cells of 50 by 50 kms over the European continent and marking the occurences of each species per cell. Every cell therefore generates a transaction with boolean items representing the presence/absence of a particular species in a given area. The data can be found in file `bats.mat` while metadata (id and coordinates) for each cell can be found in file `bats_coords.csv`. Information about the general study/efforts from which this dataset is taken can be found on the official website of the European Mammal Foundation. Finding patterns in this dataset can reveal information on the natural habitats of different species and the possibility of their coexistence. This information can serve environmental and biological studies focused on animal behavior.

Loading data This dataset is loaded using function `load_matrix` (l:108). Said function generates transactions from the input file `bats.mat` by saving the index of the species type present in every region. Every transaction represents a single region, with items equal to the indices of

the species present in it. Regions where no bats are observed are added as empty transactions. The value space of the items (U) is the range of indices of the available species (from 0 to 23).

1.2.3 *Retail* - basket data

Description This dataset contains transactions from a retail store in Belgium collected in three different periods between December 1999 and November 2000 [2]. The data is in the form of basket data with each item corresponding to the `id` of a particular purchased article, and each line representing a given transaction. Studying this dataset is likely to reveal patterns in the purchasing behavior of customers of the given store. This kind of analysis (known as *basket analysis*) is credited for the launching of the field of frequent itemset mining.

Loading data Generating the transactions database is therefore straightforward: all we have to do is read the items from the file `retail.dat` and cast them as integers. This is done by the function `load_trans_num()` (l: 88). The resulting data is stored in file `X_retail.dat`.

1.2.4 *Groceries* - shopping data

Description This data contains transactions made by customers in a given store (there is no information about the store in the data source). The dataset is given in file `groceries.csv`. Every line of the file represents a single transaction. Studies on this dataset would have similar aims as described for retail dataset.

Loading data Items in this case are in text format, and need to be assigned numbers representing their unique value. This is done by putting all items in a set, then enumerating said set to generate a dictionary of item-index tuples, which is used to reconstruct the dataset with each transaction expressed as a list of such tuples. The code for loading the data be found in function `load_trans_txt()` (l: 100). The parsed transactions are in file `X_groceries.dat`.

1.2.5 *Adult* - income prediction

Description This dataset contains personal information on individual participants alongside their annual income (a description of all the attributes can be found in `adult.names`). The goal of the set is to predict the income of an individual based on the other attributes given. Frequent patterns in this dataset can reveal the personal features that influence a person's chances of making a high income.

Loading data Continuous attributes were discretized in a similar manner to *abalone*. The resulting categorical values can be found in `adult.bininfo`, and the data loading is performed using function `load_data_txt()` to result in file `X_adult.dat`.

1.2.6 *Accidents* - traffic accidents

Description The data shows the traffic accidents recorded in Belgium between 1991 and 2000. The dataset is in the form of a series of transactions with numerical values. Each line (transaction) represents one accident, while each item is a value assigned to a specific accident for a descriptive attribute used in the records. There are over 50 attributes for describing the accidents (columns of the data), with each record providing values for around 45 of them (a detailed record of the attributes can be found in [3]). Frequent patterns from this data can reveal interesting features of traffic road accidents, such as where they are most likely to happen, in which weather conditions, by what type of driver etc. This is likely to inform decision makers on where to focus their efforts for preventing similar accidents in the future.

Loading data The data provided in the assignment appears to be pre-processed, and is therefore simply read into a matrix, similarly to *Retail*.

1.2.7 *Abalone* - sea snails' data

Description The data here represents different attributes of a sea snail species known as abalone. The attributes include weight, length, diameter, and height (the complete list of attributes can be found in `abalone.bininfo`). Each transaction in the data represents characteristics of an individual abalone, while items are values of said characteristics. Frequent patterns on this dataset can reveal typical characteristics of the abalone and its potential subspecies.

Loading data The file given with the assignment material (`abalone.data`) contains continuous values for its attributes. These values are discretized using information provided in `abalone.bininfo` to give categorical attributes, which can then be loaded in a similar manner to *House* dataset (see section Loading data under section 1.2.1). The resulting file can be found in `X_abalone.dat`.

1.3 Results

Overall, we can say that `eclat` and `fpgrowth` behave similarly, with no significant difference in execution time. In terms of dataset characteristics, we notice three different trends in execution time: near constant execution (datasets *Retail*, *Adult*, and *Groceries*), exponential decrease (*House*, *Bats*, and *Abalone*), and slow execution with minor decrease (*Accidents*). The figures in appendices E.1, F.1, and G.1 were arranged to show these groupings. The first plot of each row is representative of its trend (i.e., showcases it best).

Datasets with near constant execution times have two properties in common: low density and small/constant transaction length. Density can be best seen in the binary matrix plots in appendix B. *Retail*, the dataset with the lowest density level reported at 0.0006, incidently has the execution time least affected by changes in support levels for mining closed and maximal itemsets. For both of these patterns, the range of possible support values is of medium value

(1 to around 46), meaning the most frequent itemsets appear in about half the transactions of the dataset. The quantity of mined sets for *Retail* is the lowest among datasets in all three patterns (159 closed itemsets, 78 maximal sets, and 931 association rules), with generally small sets' length (the average closed itemset length at support level 1 was 1.7). Datasets *Adult* and *Groceries* display similar characteristics. *Adult* has a wider support interval range and longer mined sets on average. Most notably, on mining association rules which have an upper bound on support level of around 1 for both *Groceries* and *Retail*, *Adult* returns association rules of near 95 frequency. This can be explained by the higher density of *Adult* compared to the other two datasets. *Groceries* the smallest upper bound on support level, which is due to its small average transaction length.

Exponential decrease in execution time is witnessed in datasets *House*, *Bats*, and *Abalone*. *House*, which is the dataset with the largest density, appears to be most affected by changes in support levels. We can see a drastitc drop in execution time and the size of the result set at around support value 15%. Although the number of mined itemsets is high at low support values, only one item seems to cover around half the dataset, which is comparable to *Retail*. This is due to the relatively small size of dataset *House* (with 435 transactions, it is by large the smallest dataset explored). The length of the mined itemsets seems larger than those found in the previous datasets' group, although it quickly drops with higher support values. Finally, the range of support values possible for association rules is almost the same as for closed and maximal itemsets, which is to be expected given the high density. *Bats* and *Abalone* show a similar behavior. *Abalone* appears to have a smaller upper bound on support levels, which can be explained by its medium (and constant) transaction length.

The last category of datasets comprises *Accidents* alone. Despite its relatively low density, this dataset takes the longest time to mine. It also has multiple frequent items at high support values (we still get 31 closed itemsets at support value 90). The length of the itemsets appears to be long even at high support values (e.g., at support value 100 we still get 11 association rules with an average length of 11). These results are due to the large size of the dataset ($340,183$ transactions), and the high length of transactions a maximum value of 51% and average of 33.

1.4 Conclusion

This assignment provided an opportunity to investigate the data characteristics most likely to affect the execution and running time of two data-mining algorithms: `fp-growth` and `eclat`. We identified 4 such characteristics: items' frequency, transactions' length, database density, and database size. Density seems to have the most effect on execution time: a dense database takes a longer time to mine than a large database. Transactions' length and items' frequency affect the length and quantity of mined patterns, and are particularly influential on association rules. Finally database size seems to be the characteristics least affecting the algorithms', which shows that the algorithms have been optimized to handle large datasets.

2 Task 3

2.1 *Apriori* algorithm

In this task, we are asked to implement a simple level-wise search (also known as *Apriori*) algorithm. The lecture slides explain how this algorithm works. In essence, *Apriori* generates all possible itemsets of length k, prunes out the ones with a frequency below the support threshold, and merges the remaining ones to create itemsets of length $k+1$ and so on. This process carries on until no more frequent itemsets can be found. This approach is guaranteed to work thanks to the monotonicity property of frequency. Figure 3 shows the pseudo-code provided in [4].

The implementation of this algorithm can be found in function `apriori()` (`miner.py`, l:79). The code expects as input the name of a dataset (see the file `helper.py` for suggestions), and generates as output a detailed report of the candidates considered by *Apriori* at every level, as well as the output of the `eclat` algorithm from `fim` for comparison. To run this code, use the command: `python miner.py -d dataset -s support -c confidence`.

Algorithm 2 Apriori Frequent Itemset Algorithm

INPUT: A file $\mathcal{D}$ consisting of baskets of items, a support threshold σ.
OUTPUT: A list of itemsets $\mathcal{F}(\mathcal{D}, \sigma)$.
METHOD:
$C_1 \leftarrow \{\{i\} | i \in \mathcal{J}\}$
$k \leftarrow 1$
while $C_k \neq \{\}$ **do**
# Compute the supports of all candidate itemsets
    for all transactions $\{tid, I\} \in \mathcal{D}$ **do**
        for all candidate itemsets $X \in C_k$ **do**
            if $X \subseteq I$ **then**
                $X.support$++
Extract all frequent itemsets
    $F_k = \{X | X.support > \sigma\}$
# Generate new candidate itemsets
    for all $X, Y \in Fi, X[i] = Y[i]$ for $1 \leq i \leq k-1$, and $X[k] < Y[k]$ **do**
        $I = X \cup \{Y[k]\}$
        if $\forall J \subset I, |J| = k : J \in F_k$ **then**
            $C_{k+1} \leftarrow C_{k+1} \cup I$
    k++

Figure 1: Pseudo-code for *Apriori* algorithm from [4]

2.2 Generating association rules

Association rules define implication relationships between subsets of frequent itemsets. They seek to answer the question of 'how likely is it to find item y in a transaction, given that it contains item x?'. We are generally interested in rules with a confidence above a given threshold.

The confidence of a rule is computed as shown in equation in equation (1). My implementation of association mining can be found in function `get_association_rules()` (`miner.py`, l:99). The code generates a file (`output/asso_r/dataset_support_confidence.txt`) containing the mined rules with their corresponding confidence after every run. I did not evaluate the output of my implementation against a standard implementation (e.g. `fim`'s `eclat`), since I haven't debugged my *Apriori* code yet, so I expect to be missing at least the association rules of the missing frequent itemsets.

$$conf(X \Rightarrow Y) = \frac{|supp(X \cup Y)|}{|supp(X)|} \tag{1}$$

3 Task 4

3.1 IBM Generator Data

Running the output of the suggested commands can be found under *datasets/IBM*. Each command generates 3 files, one for configuration (`.conf`), one containing the dataset (`.dat`), and one containing patterns in said data (.pat). The dataset file contains lines representing transactions. Each line starts with a transaction ID replicated twice for consistency with the generator (source, documentation). The remaining integers on the line are the items making up a transaction. The patterns' file contains general statistics on the patterns generated. I could not find information on the meaning of the first two numbers next to every itemset on the file, but I suspect they refer to the support of the itemset and a related measure.

3.2 Synthetic data

I used the functions provided in the helper code (l:36 to 86 in `helper.py`) to generate synthetic data with different properties. These properties are best seen in the binary matrix plots of the datasets, shown in appendix B. From figure 2, we can see that the only difference between the sets is the length of their transactions. This would allow us to assess the effect of the distribution of items in a dataset on frequent itemset mining results. Unfortunately, I did not get to run the algorithms on this synthetic data. I expect their performance to confirm the trends witnessed in real datasets (on tasks 1 and 2) where datasets with longer transactions took longer to mine, and resulted in longer frequent itemsets, and a higher upper bound on support value.

dataset	items	trans	density	min_trans	max_trans	avg_trans
random_U	50	1000	0.1016	0	13	5.082
random_B	50	1000	0.1	5	5	5.0
random_H	50	1000	0.1	0	30	5.0
random_V	18	1000	0.102	5	5	5.0
random_P	45	999	0.1168	0	9	5.2553

Figure 2: *Basic statistics of synthetic data*

Appendices

A General statistics

dataset	items	trans	density	min_trans	max_trans	avg_trans
retail	16470	88162	0.0006	1	76	10.3058
adult	32	32561	0.0825	8	8	8.0
groceries	169	9835	0.0259	1	32	4.4095
house	50	435	0.3333	17	17	17.0
bats	23	2575	0.1892	0	18	4.7305
abalone	43	4177	0.2045	9	9	9.0
accidents	468	340183	0.0721	18	51	33.8079

Figure 3: Descriptive statistics for all 7 datasets

B Binary Matrices

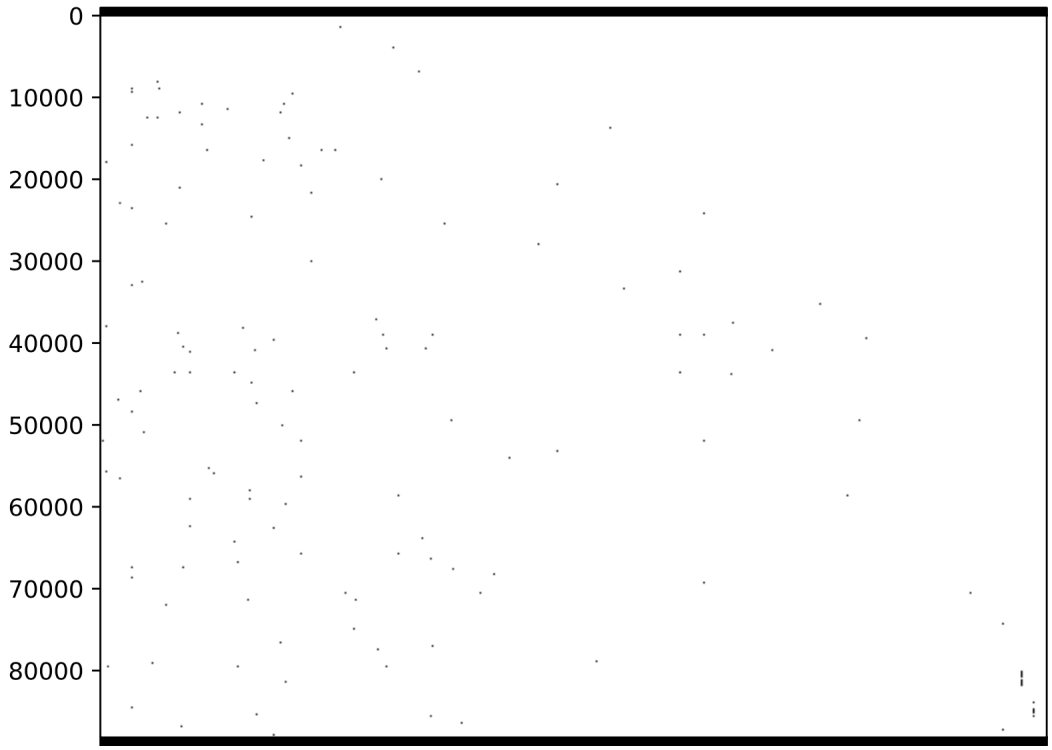

Figure 4: *Retail*

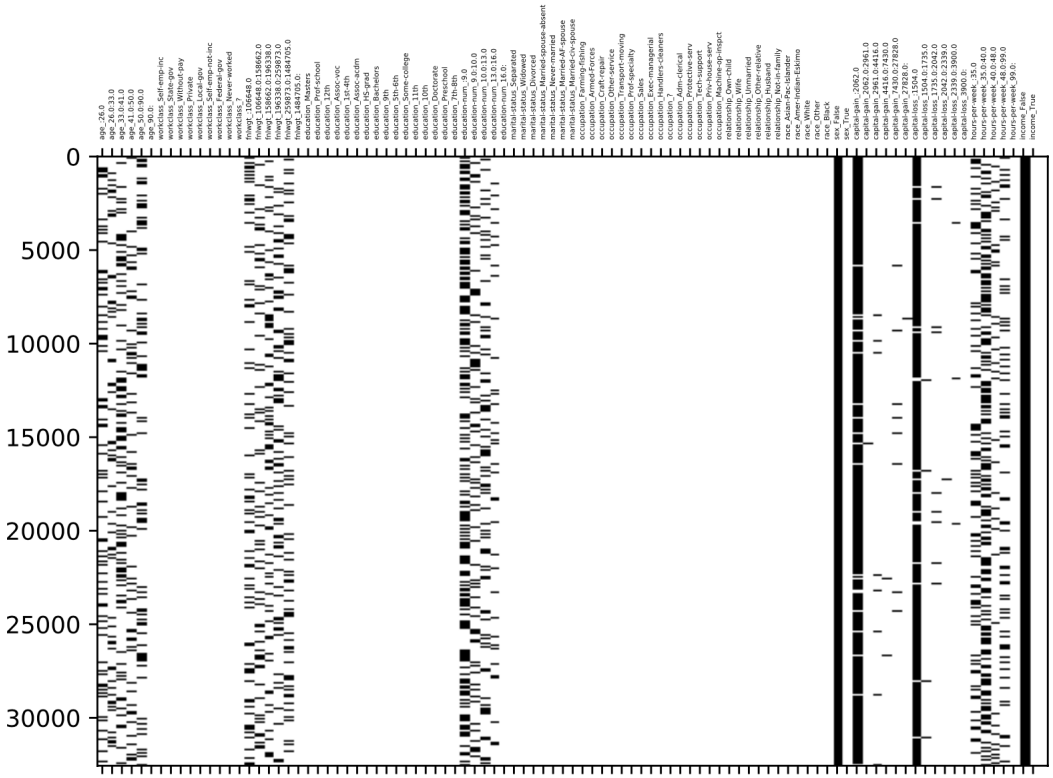

Figure 5: *Adult*

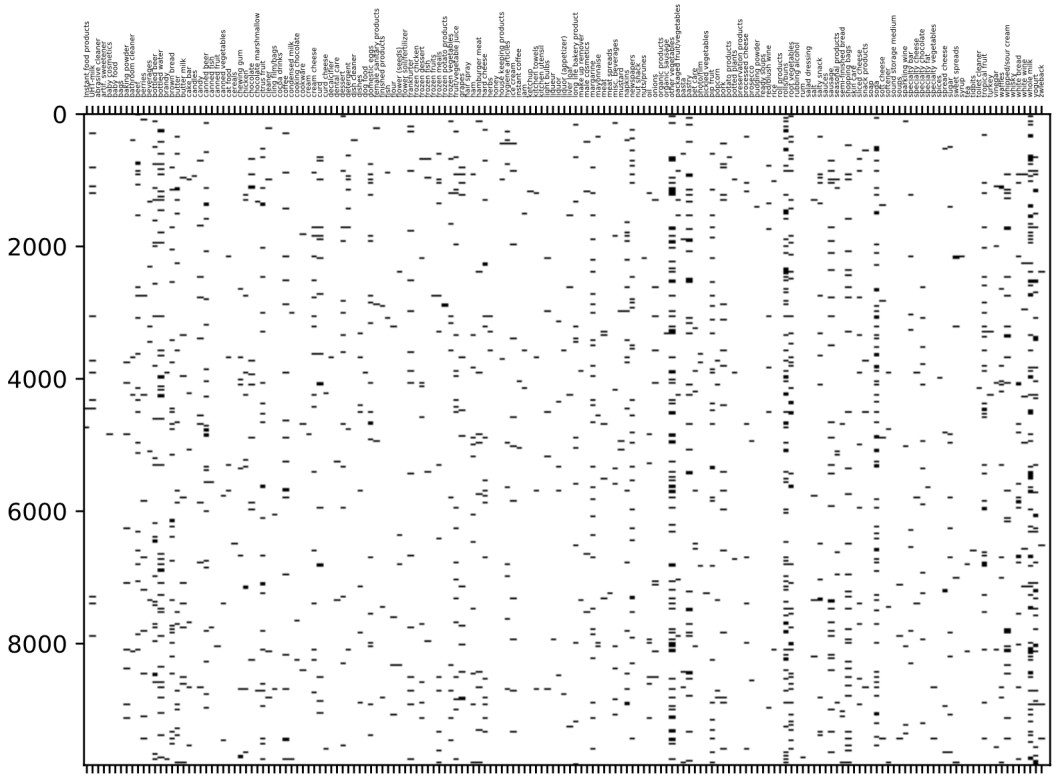

Figure 6: *Groceries*

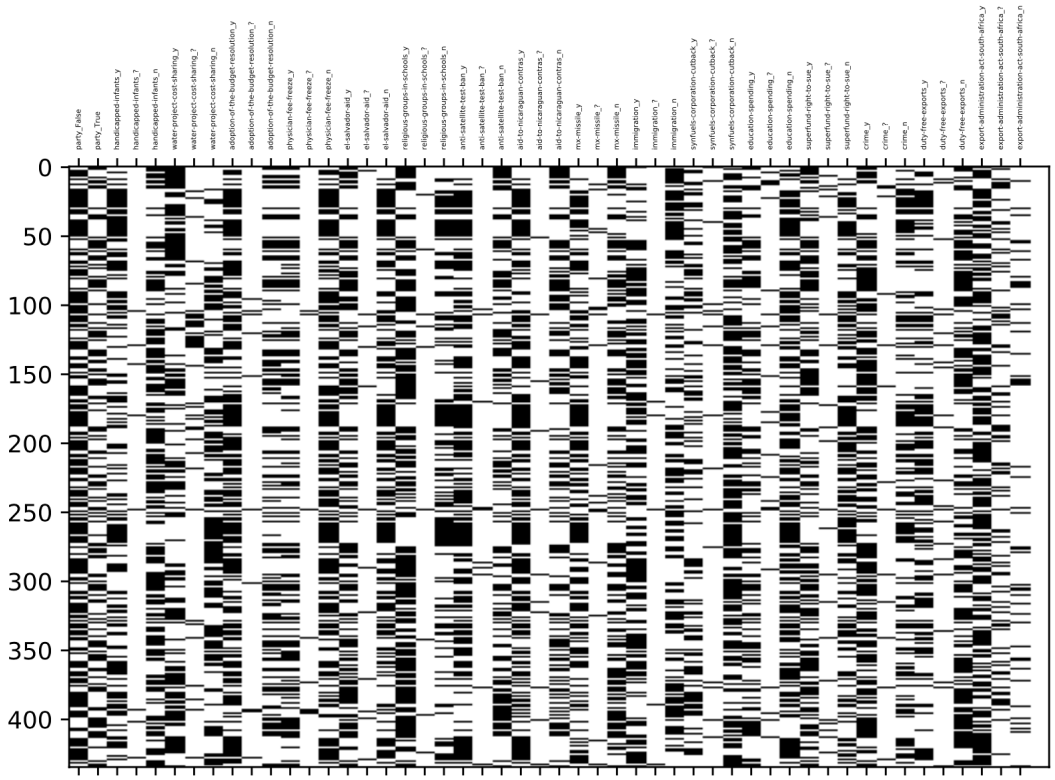

Figure 7: *House*

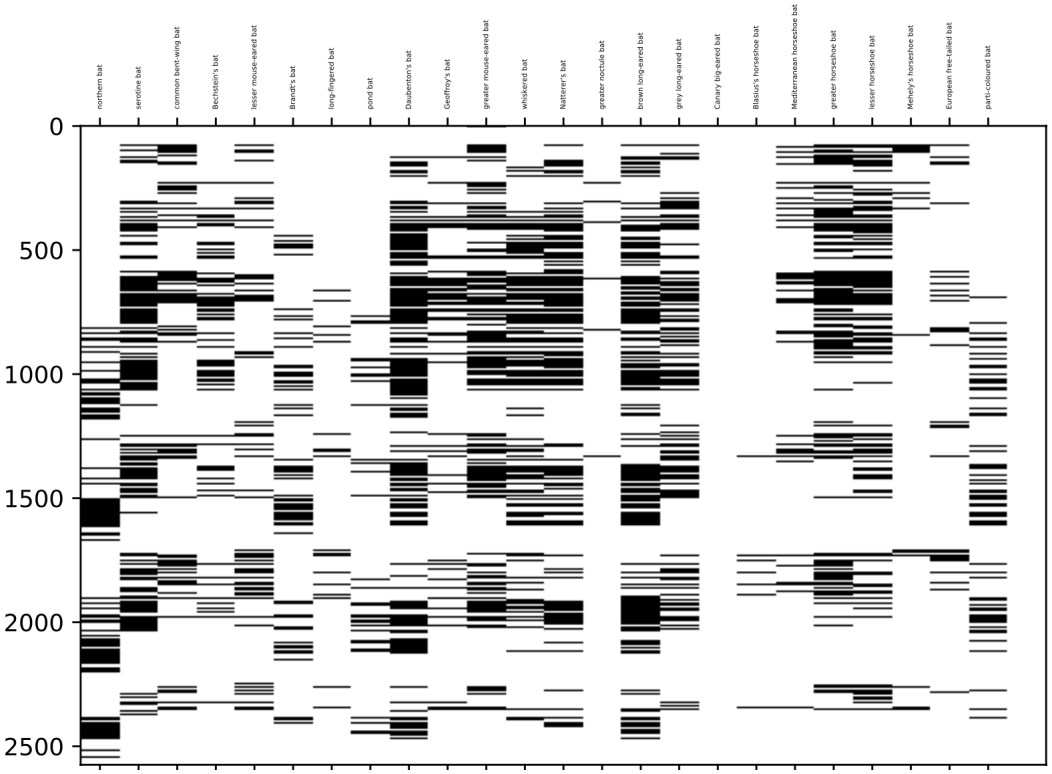

Figure 8: *Bats*

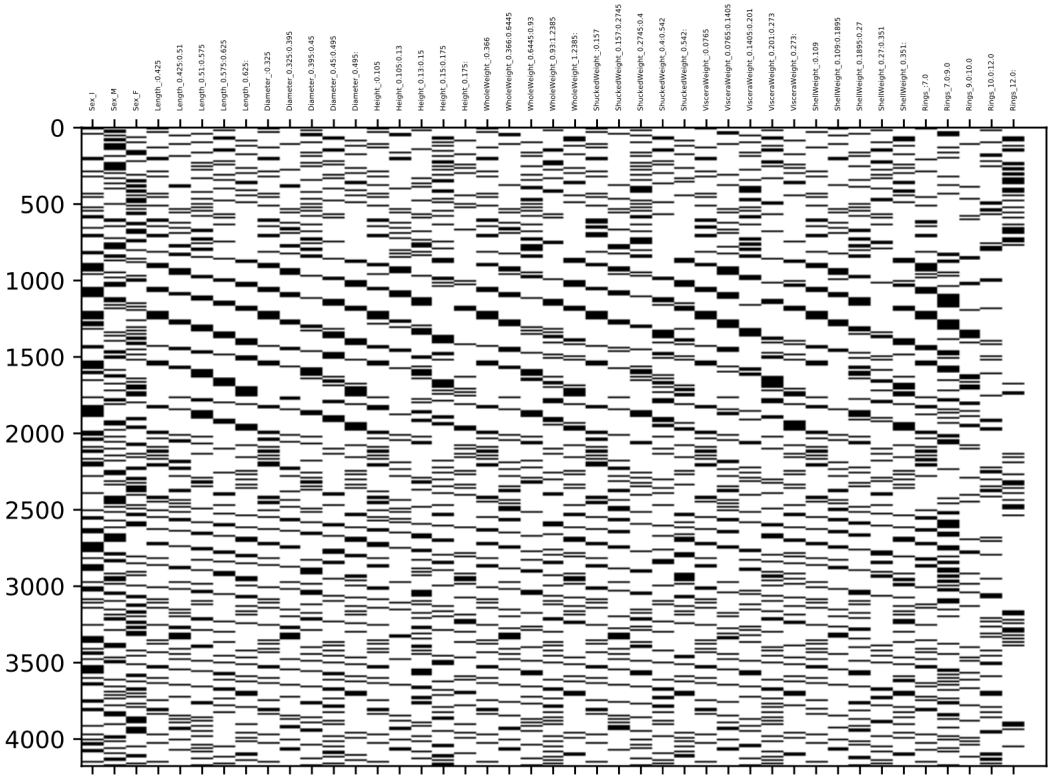

Figure 9: *Abalone*

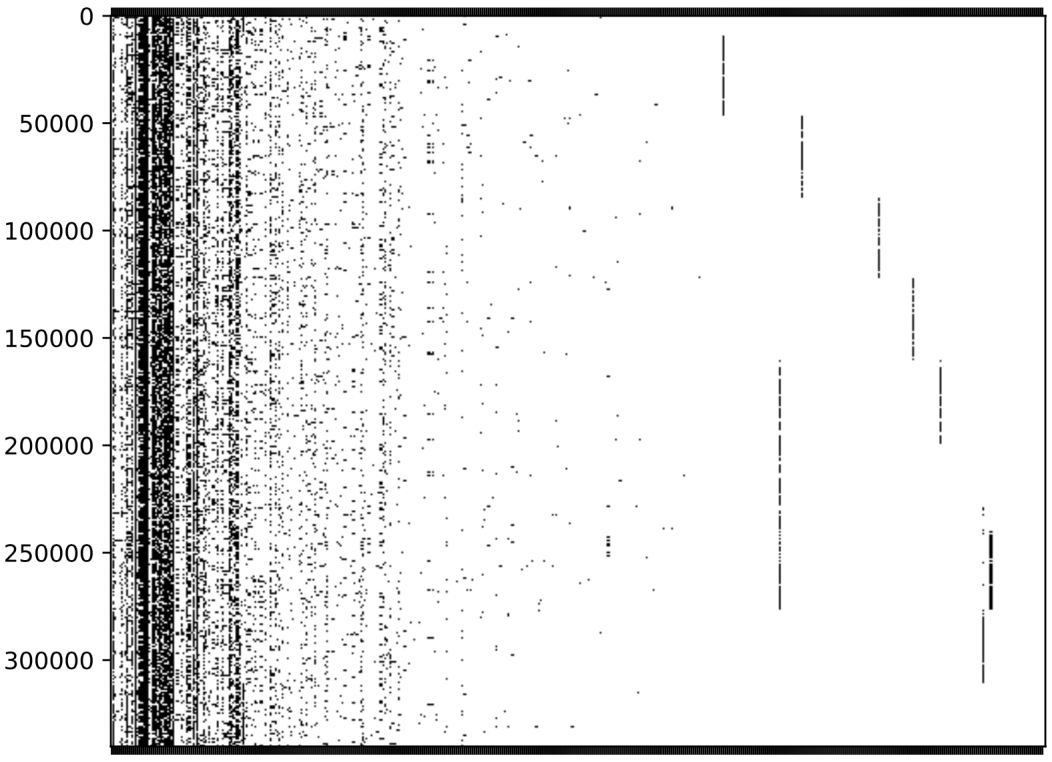

Figure 10: *Accidents*

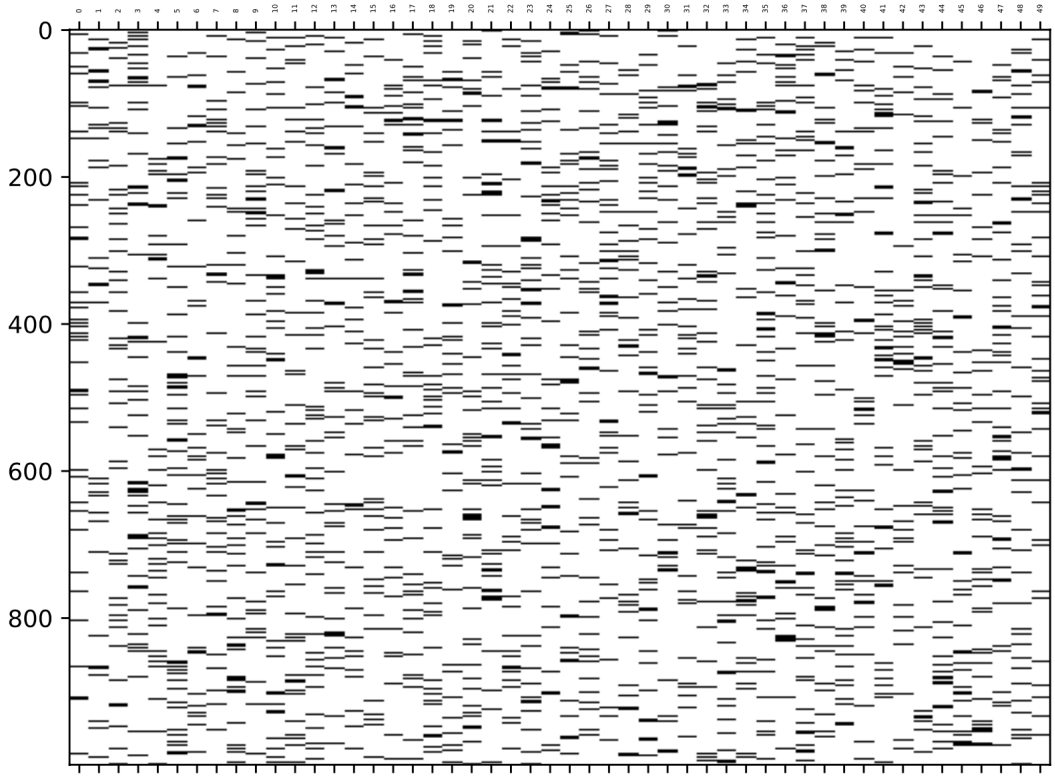

Figure 11: *Uniform random*

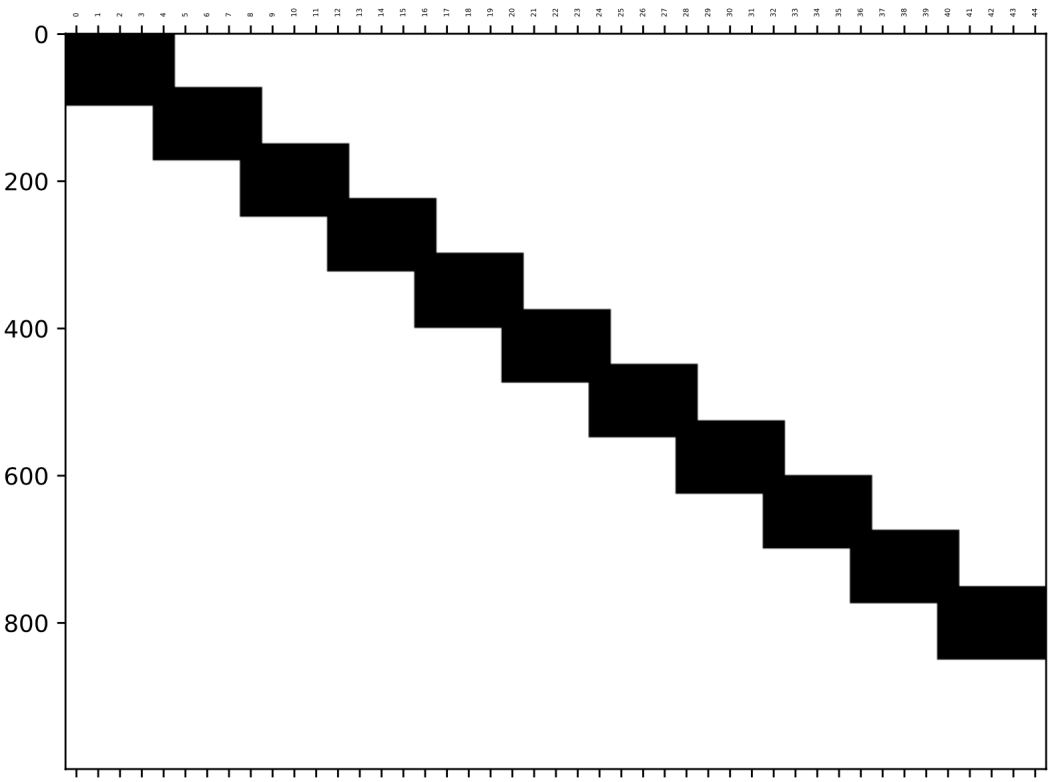

Figure 12: *Pieces random*

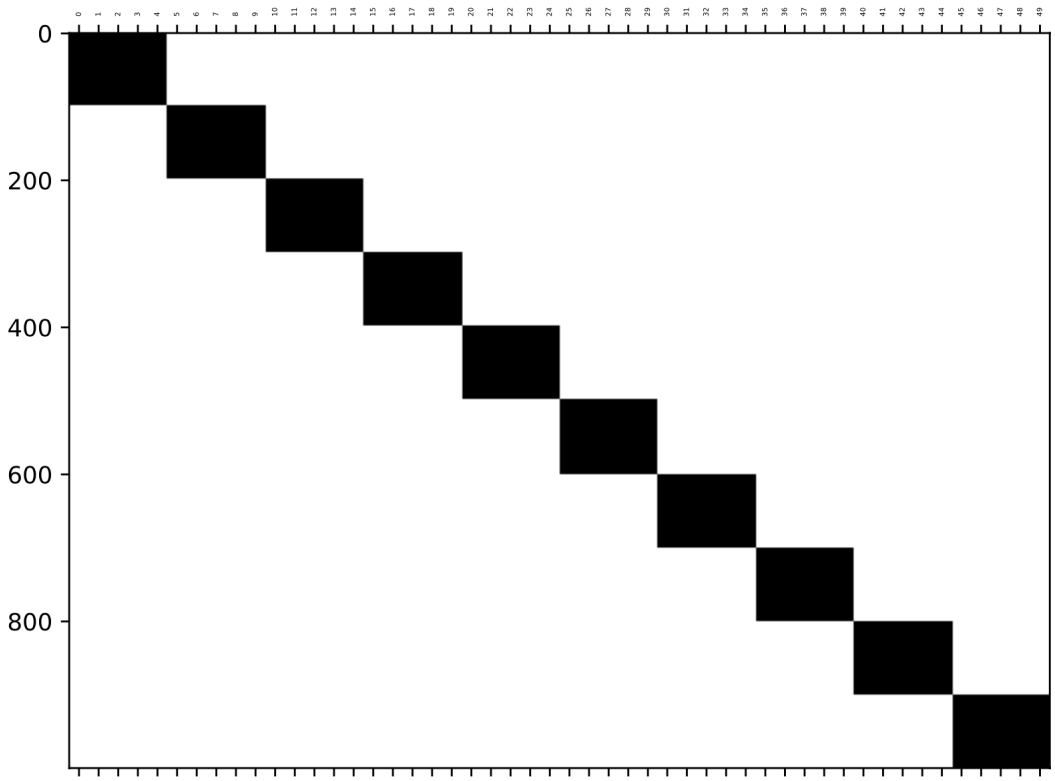

Figure 13: *Blocks random*

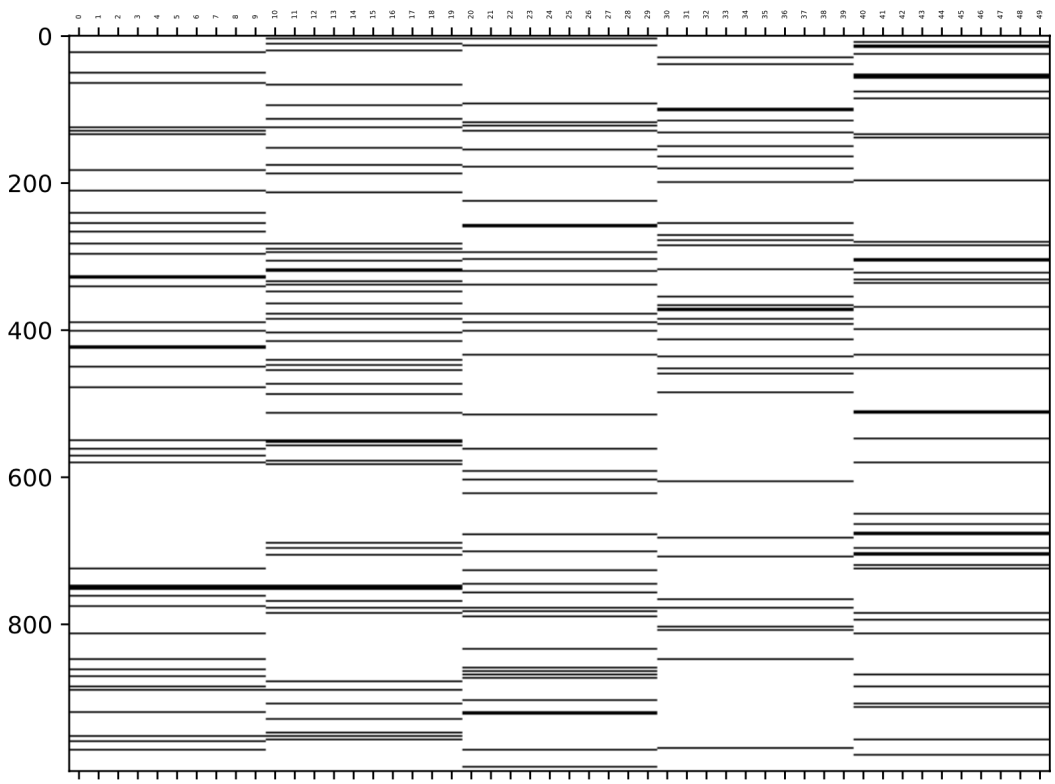

Figure 14: *Horizontal random*

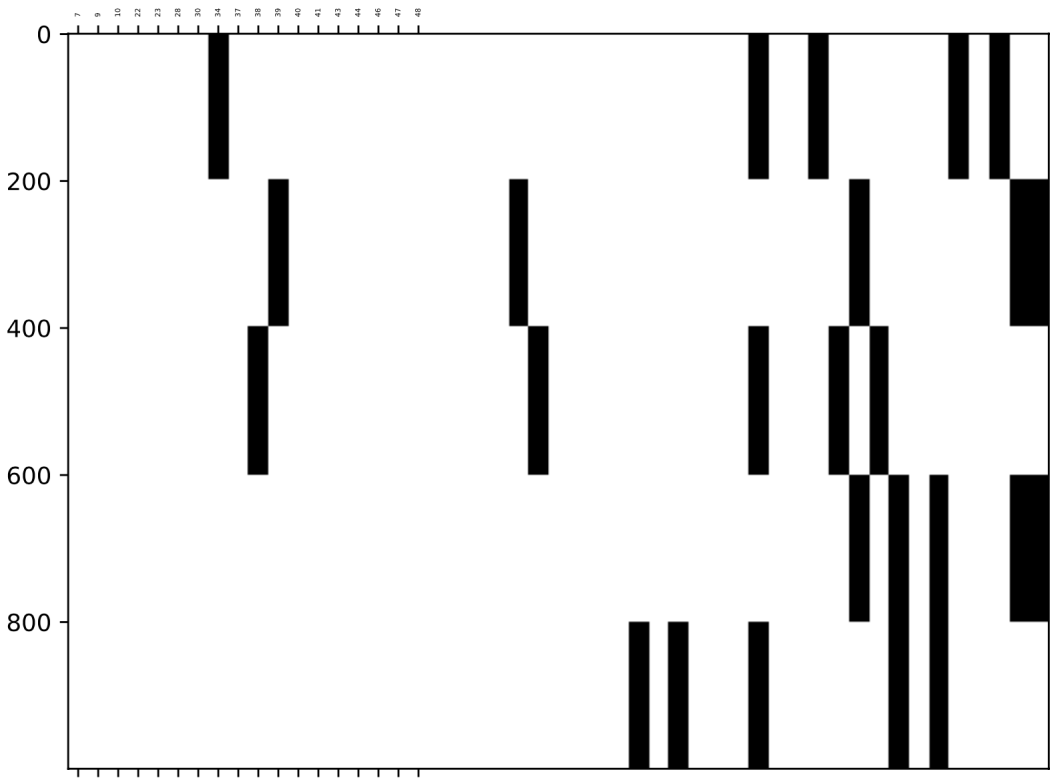

Figure 15: *Vertical random*

C Items' distribution

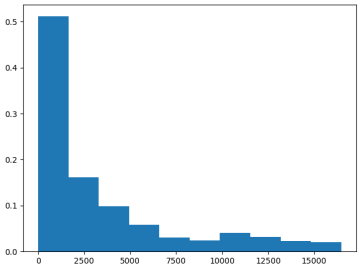

Figure 16: *Retail*

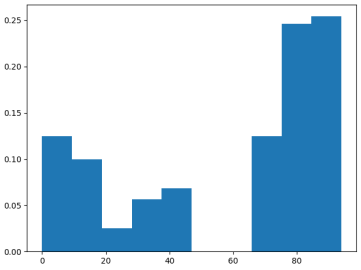

Figure 17: *Adult*

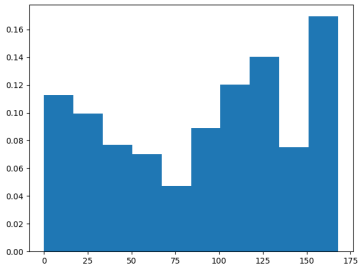

Figure 18: *Groceries*

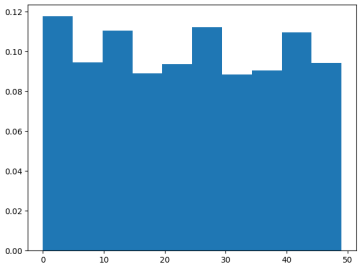

Figure 19: *House*

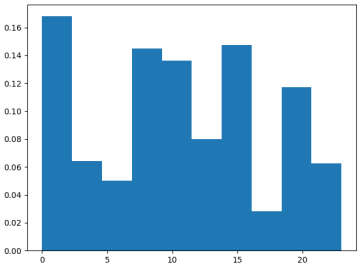

Figure 20: *Bats*

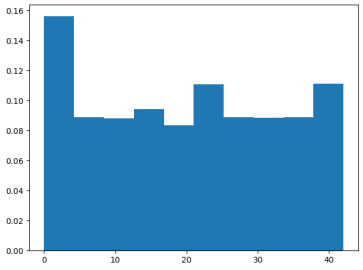

Figure 21: *Abalone*

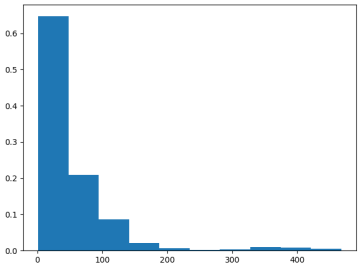

Figure 22: *Accidents*

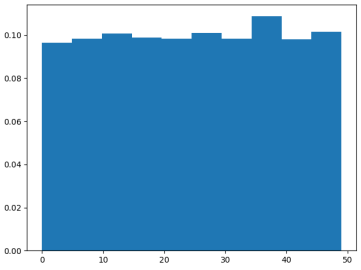

Figure 23: *Uniform random*

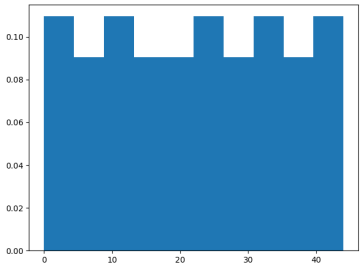

Figure 24: *Pieces random*

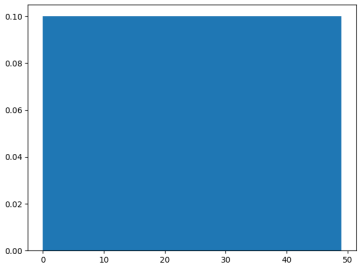

Figure 25: *Blocks random*

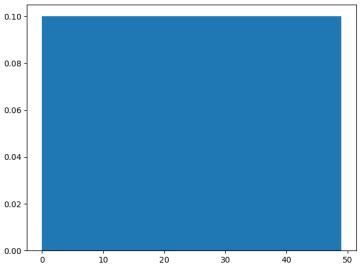

Figure 26: *Horizontal random*

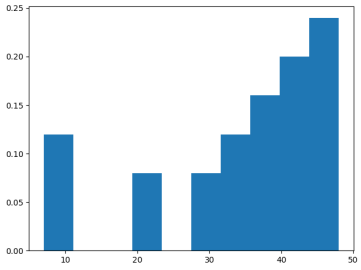

Figure 27: *Vertical random*

D Transactions' length distribution

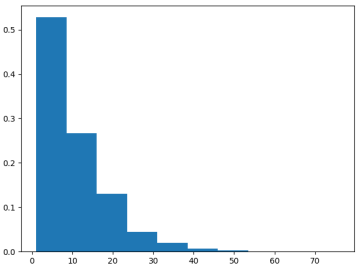

Figure 28: *Retail*

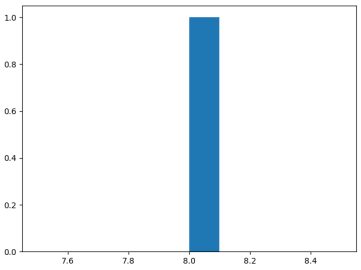

Figure 29: *Adult*

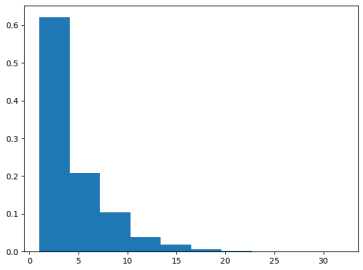

Figure 30: *Groceries*

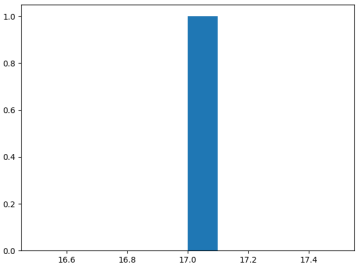

Figure 31: *House*

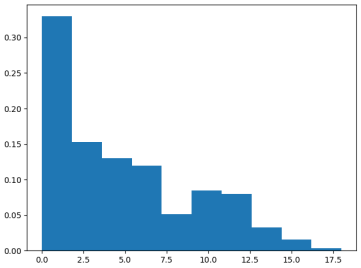

Figure 32: *Bats*

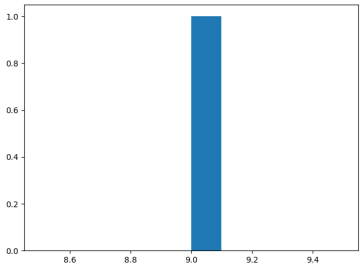

Figure 33: *Abalone*

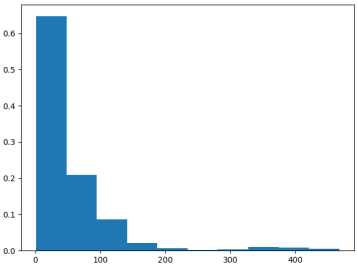

Figure 34: *Accidents*

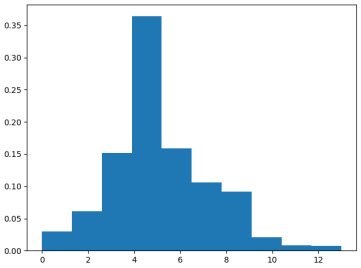

Figure 35: *Uniform random*

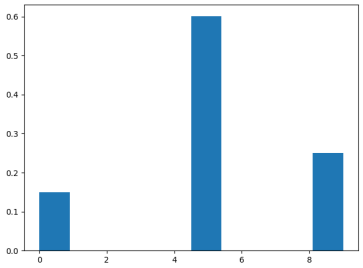

Figure 36: *Pieces random*

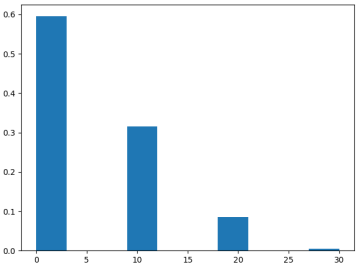

Figure 37: *Horizontal random*

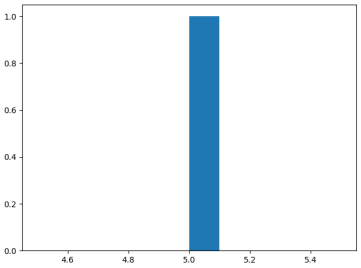

Figure 38: *Vertical random*

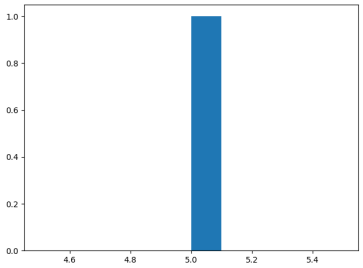

Figure 39: *Blocks random*

E Mining closed itemsets

E.1 Running times

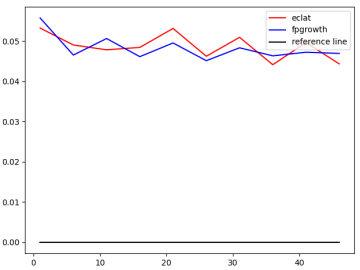

Figure 40: *Retail*

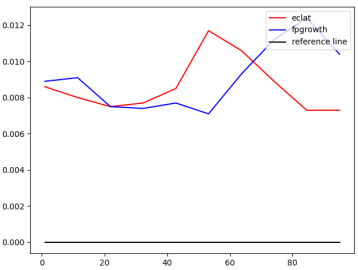

Figure 41: *Adult*

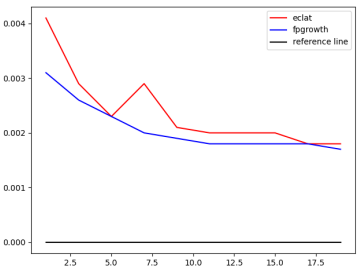

Figure 42: *Groceries*

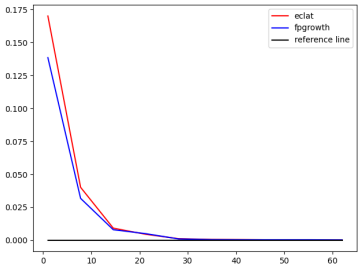

Figure 43: *House*

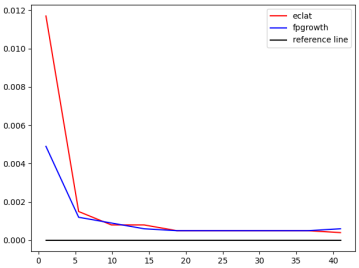

Figure 44: *Bats*

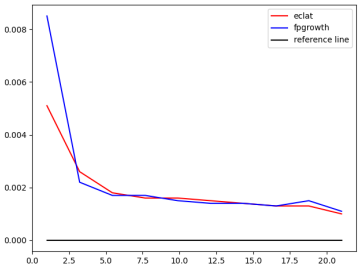

Figure 45: *Abalone*

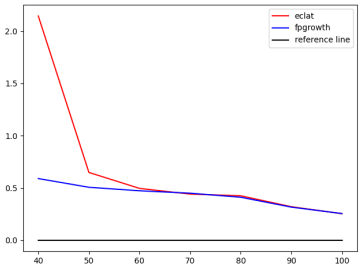

Figure 46: *Accidents*

E.2 Statistics

support	n_obs	min	max	avg	variance
1	159	1	4	1.7925	0.7098
6	15	1	3	1.8667	0.5524
11	8	1	2	1.375	0.2679
16	6	1	2	1.1667	0.1667
21	3	1	2	1.3333	0.3333
26	3	1	2	1.3333	0.3333
31	3	1	2	1.3333	0.3333
36	2	1	1	1.0	0.0
41	2	1	1	1.0	0.0
46	2	1	1	1.0	0.0

Figure 47: *Retail*

support	n_obs	min	max	avg	variance
1.0	1132	3	8	5.4337	0.9072
11.44	72	3	6	4.0278	0.5626
21.89	19	3	6	3.9474	0.8304
32.33	11	3	5	3.8182	0.5636
42.78	10	3	5	3.7	0.4556
53.22	3	3	4	3.3333	0.3333
63.67	3	3	4	3.3333	0.3333
74.11	3	3	4	3.3333	0.3333
84.56	3	3	4	3.3333	0.3333
95.0	1	3	3	3.0	0.0

Figure 48: *Adult*

support	n_obs	min	max	avg	variance
1	333	1	3	1.8318	0.3331
3	63	1	2	1.3016	0.214
5	31	1	2	1.0968	0.0903
7	19	1	2	1.0526	0.0526
9	10	1	1	1.0	0.0
11	6	1	1	1.0	0.0
13	5	1	1	1.0	0.0
15	4	1	1	1.0	0.0
17	4	1	1	1.0	0.0
19	2	1	1	1.0	0.0

Figure 49: *Groceries*

support	n_obs	min	max	avg	variance
1.0	234629	1	17	7.4091	4.8642
7.78	66023	1	15	6.8968	4.1704
14.56	21022	1	13	6.1486	3.8906
21.33	7768	1	10	5.1973	2.3026
28.11	1368	1	8	3.6637	1.4743
34.89	314	1	6	2.9076	1.3045
41.67	79	1	4	2.0759	0.8403
48.44	23	1	3	1.3913	0.3399
55.22	9	1	2	1.1111	0.1111
62.0	1	1	1	1.0	0.0

Figure 50: *House*

support	n_obs	min	max	avg	variance
1.0	15811	1	13	6.2378	3.4335
5.44	2147	1	9	4.6656	1.9375
9.89	421	1	7	3.4703	1.5211
14.33	124	1	5	2.621	0.969
18.78	43	1	4	2.1163	0.6766
23.22	21	1	3	1.619	0.4476
27.67	10	1	2	1.3	0.2333
32.11	6	1	2	1.1667	0.1667
36.56	2	1	1	1.0	0.0
41.0	2	1	1	1.0	0.0

Figure 51: *Bats*

support	n_obs	min	max	avg	variance
1.0	8467	1	9	4.1424	1.5746
3.22	2365	1	9	3.7852	1.9039
5.44	1296	1	9	3.7307	2.1043
7.67	913	1	9	3.7689	2.4981
9.89	726	1	9	3.9187	2.7175
12.11	527	1	8	3.6698	2.3737
14.33	216	1	7	2.7917	1.961
16.56	133	1	6	2.5038	1.6761
18.78	43	1	2	1.1163	0.1052
21.0	8	1	1	1.0	0.0

Figure 52: *Abalone*

support	n_obs	min	max	avg	variance
40	32528	1	11	5.765	2.0217
50	8057	1	9	4.9969	1.7871
60	2074	1	8	4.2816	1.5637
70	529	1	7	3.6427	1.4195
80	149	1	6	3.094	1.2749
90	31	1	5	2.5806	1.1183

Figure 53: *Accidents*

F Mining maximal itemsets

F.1 Running times

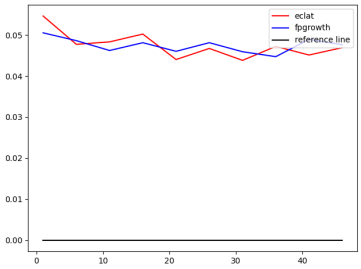

Figure 54: *Retail*

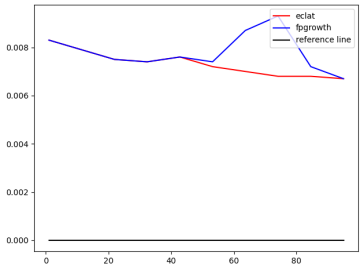

Figure 55: *Adult*

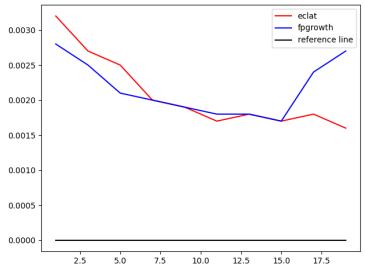

Figure 56: *Groceries*

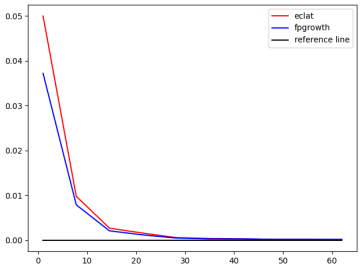

Figure 57: *House*

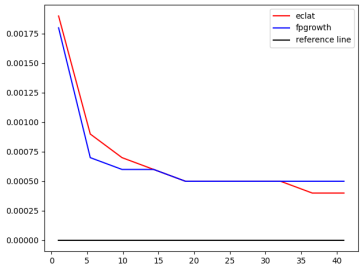

Figure 58: *Bats*

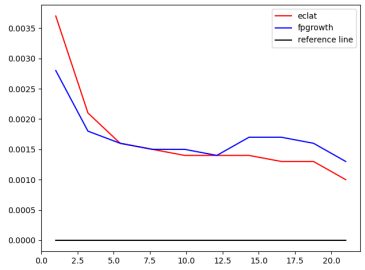

Figure 59: *Abalone*

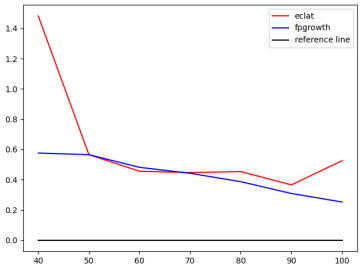

Figure 60: *Accidents*

F.2 Statistics

support	n_obs	min	max	avg	variance
1	78	1	4	1.7308	0.8487
6	3	3	3	3.0	0.0
11	4	1	2	1.75	0.25
16	4	1	2	1.25	0.25
21	1	2	2	2.0	0.0
26	1	2	2	2.0	0.0
31	1	2	2	2.0	0.0
36	2	1	1	1.0	0.0
41	2	1	1	1.0	0.0
46	2	1	1	1.0	0.0

Figure 61: *Retail*

support	n_obs	min	max	avg	variance
1.0	193	4	8	6.5233	0.5424
11.44	16	4	6	5.0	0.1333
21.89	4	3	6	4.0	2.0
32.33	2	5	5	5.0	0.0
42.78	3	4	5	4.3333	0.3333
53.22	1	4	4	4.0	0.0
63.67	1	4	4	4.0	0.0
74.11	1	4	4	4.0	0.0
84.56	1	4	4	4.0	0.0
95.0	1	3	3	3.0	0.0

Figure 62: *Adult*

support	n_obs	min	max	avg	variance
1	243	1	3	1.963	0.3003
3	50	1	2	1.38	0.2404
5	27	1	2	1.1111	0.1026
7	17	1	2	1.0588	0.0588
9	10	1	1	1.0	0.0
11	6	1	1	1.0	0.0
13	5	1	1	1.0	0.0
15	4	1	1	1.0	0.0
17	4	1	1	1.0	0.0
19	2	1	1	1.0	0.0

Figure 63: *Groceries*

support	n_obs	min	max	avg	variance
1.0	15435	2	17	8.1372	4.3317
7.78	3142	1	15	7.2269	6.7365
14.56	1088	2	13	5.6783	4.3619
21.33	826	1	10	6.0036	3.257
28.11	268	2	8	4.0858	1.6143
34.89	74	1	6	3.2568	1.7003
41.67	32	1	4	2.2188	1.1442
48.44	14	1	3	1.5	0.4231
55.22	7	1	2	1.1429	0.1429
62.0	1	1	1	1.0	0.0

Figure 64: *House*

support	n_obs	min	max	avg	variance
1.0	500	2	13	7.15	3.6187
5.44	221	1	9	5.6561	2.3721
9.89	85	2	7	3.8706	1.3759
14.33	42	1	5	3.0238	1.0482
18.78	14	1	4	2.5	0.7308
23.22	10	1	3	1.8	0.6222
27.67	7	1	2	1.4286	0.2857
32.11	4	1	2	1.25	0.25
36.56	2	1	1	1.0	0.0
41.0	2	1	1	1.0	0.0

Figure 65: *Bats*

support	n_obs	min	max	avg	variance
1.0	1529	2	9	4.2184	1.2114
3.22	375	2	9	3.4773	1.9293
5.44	160	2	9	3.2625	1.4653
7.67	109	1	9	2.7982	1.2181
9.89	59	1	9	2.661	1.6762
12.11	47	1	8	2.7021	2.3876
14.33	55	1	7	1.9818	1.0552
16.56	39	1	6	1.6923	1.6923
18.78	38	1	2	1.1316	0.1174
21.0	8	1	1	1.0	0.0

Figure 66: *Abalone*

support	n_obs	min	max	avg	variance
40	762	4	11	7.9843	1.3848
50	216	3	9	7.0509	1.3974
60	78	3	8	6.2308	0.985
70	23	4	7	5.4783	0.8063
80	8	3	6	4.625	1.4107
90	1	5	5	5.0	0.0

Figure 67: *Accidents*

G Mining association rules

G.1 Running times

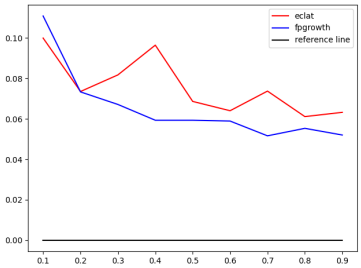

Figure 68: *Retail*

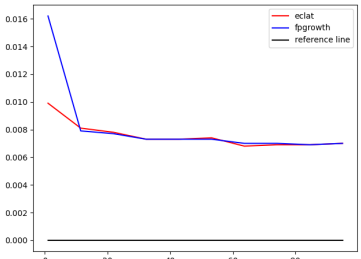

Figure 69: *Adult*

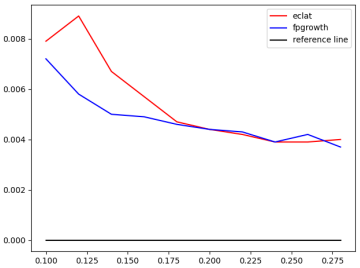

Figure 70: *Groceries*

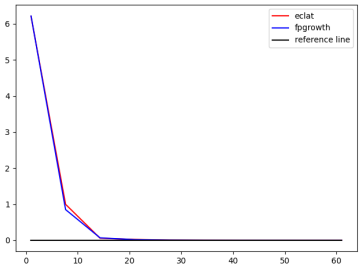

Figure 71: *House*

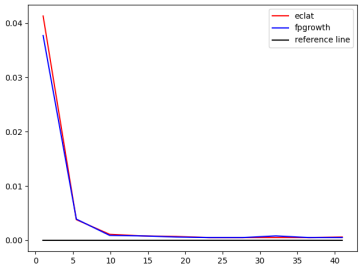

Figure 72: *Bats*

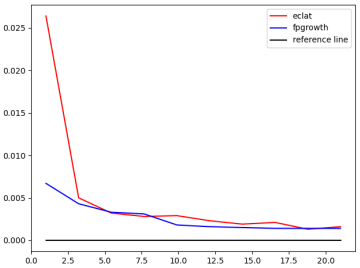

Figure 73: *Abalone*

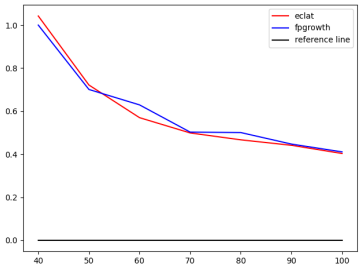

Figure 74: *Accidents*

G.2 Statistics

support	n_obs	min	max	avg	variance
0.1	931	16	32	22.8303	4.9153
0.2	272	16	29	22.0074	5.1881
0.3	150	16	27	21.6333	6.1398
0.4	98	16	27	21.3469	6.3114
0.5	69	15	27	20.8986	5.5631
0.6	55	16	26	20.7455	4.7118
0.7	41	15	24	20.3415	4.1805
0.8	29	17	24	20.2069	3.67
0.9	21	16	24	20.3333	4.8333

Figure 75: *Retail*

support	n_obs	min	max	avg	variance
1.0	9220	12	34	25.052	10.8203
11.44	584	12	29	20.8853	10.1017
21.89	168	12	29	20.9702	11.8494
32.33	92	12	26	20.4674	10.3616
42.78	88	12	26	20.2841	9.9528
53.22	28	11	23	18.1071	10.9881
63.67	28	11	23	18.1071	10.9881
74.11	28	11	23	18.1071	10.9881
84.56	28	11	23	18.1071	10.9881
95.0	16	11	23	16.6875	11.1625

Figure 76: *Adult*

support	n_obs	min	max	avg	variance
0.1	954	19	34	25.7369	8.0094
0.12	338	19	34	25.7899	7.2169
0.14	192	19	34	25.625	7.3141
0.16	106	19	33	25.0849	6.6118
0.18	50	19	30	24.58	8.371
0.2	36	19	30	24.4167	9.85
0.22	15	20	30	25.0	12.2857
0.24	11	20	30	25.5455	15.2727
0.26	8	20	30	24.75	17.6429
0.28	6	20	29	23.8333	18.1667

Figure 77: *Groceries*

support	n_obs	min	max	avg	variance
1.0	12094008	14	60	36.4721	28.6068
7.67	1115304	14	55	33.3519	24.9722
14.33	217967	14	52	30.721	25.4957
21.0	76364	14	43	28.5952	16.7683
27.67	12429	14	36	24.4181	9.7365
34.33	2067	14	31	22.0116	8.3706
41.0	451	16	27	20.3525	5.4777
47.67	93	16	22	18.2796	2.7471
54.33	24	16	19	16.9583	0.8243
61.0	3	15	16	15.6667	0.3333

Figure 78: *House*

support	n_obs	min	max	avg	variance
1.0	156551	14	46	30.1585	18.1895
5.44	8756	14	37	26.074	12.5496
9.89	1218	14	33	22.9343	9.4747
14.33	292	14	27	21.2055	7.2016
18.78	58	16	24	19.6034	3.226
23.22	20	16	22	18.15	2.7658
27.67	8	16	19	16.75	1.3571
32.11	4	15	16	15.75	0.25
36.56	2	15	16	15.5	0.5
41.0	2	15	16	15.5	0.5

Figure 79: *Bats*

support	n_obs	min	max	avg	variance
1.0	15449	16	37	25.5855	11.0674
3.22	5735	16	36	24.2942	11.4431
5.44	3620	16	36	24.0003	11.9461
7.67	2835	16	36	24.0427	13.7205
9.89	2469	16	36	24.1871	14.3159
12.11	1896	16	36	23.6292	12.9807
14.33	689	16	31	21.1814	9.1981
16.56	409	15	30	21.0391	8.2338
18.78	89	15	20	17.3596	1.3465
21.0	6	16	17	16.8333	0.1667

Figure 80: *Abalone*

support	n_obs	min	max	avg	variance
40	281826	12	47	30.2935	17.6029
50	68471	12	41	28.1275	15.9636
60	17299	12	38	26.1083	14.1408
70	4343	12	35	24.2906	12.8004
80	1243	12	32	22.72	11.9972
90	304	12	29	21.1875	12.4565
100	11	11	12	11.9091	0.0909

Figure 81: *Accidents*

Collaboration

No collaborators

References

[1] B. Christian, “Christian borgelt’s web pages,” 2019.

[2] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, “Using association rules for product assortment decisions: A case study,” in *Knowledge Discovery and Data Mining*, pp. 254–260, 1999.

[3] K. Geurts, G. Wets, T. Brijs, and K. Vanhoof, “Profiling high frequency accident locations using association rules,” in *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16*, p. 18pp, 2003.

[4] J. Heaton, “Comparing dataset characteristics that favor the apriori, eclat or fp-growth frequent itemset mining algorithms,” 01 2017.