“Frequent Itemsets, Clustering, Advertising”

Roberto Espinoza Valenzuela
Postgraduate student
A1759493

Mining Big Data
School of Computer Science
Faculty of Engineering, Computer & Mathematical Sciences
University of Adelaide

Table of contents

1. Frequent Itemsets

1.1 Simple Randomized Algorithm Implementation

Since generally speaking the sample is much smaller than the full dataset, we still avoid most of the disk I/O's that the algorithms discussed in the book and lectures, such as A-priori algorithm or Park, Chen, and Yu algorithm. Having selected our sample of the baskets. The balance of the main memory is used to execute A-Priori algorithm.

Three hash tables are used for data representation, one for each frequent item size (singletons, doubletons, tripletons), where the keys are either the pairs of ints or tuples, and the key is the count of that item. During the summary statistics phase the hash table is iterated through and every element with a count higher than the support threshold is saved.

In order to analyze the results, a script shell is executed for each algorithm. Through this script, singletons, doubletons, tripletons, runtime and memory used is showed in a log file for each sample size and support threshold, moreover, effective support threshold is showed.

1.2 Savasere, Omiecinski and Navathe algorithm implementation

In this implementation, instead of sampling a single piece of the dataset, the dataset is split into chunks and then get the union of the candidate set. Three hash tables are used for data representation, one for each frequent item size (singletons, doubletons, tripletons). Summary statistics phase is changed a little bit here, where after each chunk the hash table is reset once the items with enough support are saved.

In order to analyze the results, a script shell is executed for each algorithm. Through this script, singletons, doubletons, tripletons, runtime and memory used for each algorithm is showed in a log file for each sample size and support threshold, moreover, effective support threshold is showed.

1.3 Comparison of two algorithms on datasets

Each algorithm can be executed by a script, this script saved and outputted the following statistics:

Frequent items
Support (%)
Support threshold count
Effective support threshold count (Sample or chunk * Support threshold count)
Sample or chunk percentage
number of singletons
number of doubletons
number of tripletons
runtime in milliseconds
memory size KB used in the algorithm

Each algorithm has their own script file for each dataset

chess Randomized--------- chess.sh
connect Randomized --------- connect.sh
mushroom Randomized ------ mushroom.sh
pumsb Randomized ------- pumsb.sh
pumsb star Randomized ------- pumsb_star.sh

T10I4D100K Randomized ------ T10I4D100K.sh
T40I10D100K Randomized ------ T40I10D100K.sh
chess Son--------- chess_son.sh
connect Son --------- connect_son.sh
mushroom Son ------ mushroom_son.sh
pumsb Son ------- pumsb_son.sh
pumsb star Son ------- pumsb_star_son.sh
T10I4D100K Son ------ T10I4D100K_son.sh
T40I10D100K Son ------ T40I10D100K_son.sh

The output of each script is saved for each sample or chunk size in one single directory in the following way.

chess Randomized--------- /outputchess
connect Randomized --------- /outputconnect
mushroom Randomized ------ /outputmushroom
pumsb Randomized ------- /outputpumsb
pumsb star Randomized ------- /outputpumsb_star
T10I4D100K Randomized ------ /outputT10I4D100K
T40I10D100K Randomized ------ /outputT40I10D100K
chess Son--------- /outputchess_son
connect Son --------- /outputconnect_son
mushroom Son ------ /outputmushroom_son
pumsb Son ------- /outputpumsb_son
pumsb star Son ------- /outputpumsb_star_son
T10I4D100K Son ------ /outputT10I4D100K_son
T40I10D100K Son ------ /outputT40I10D100K_son

Every folder in that directory has the output for each sample/chunk size, 0.2 (20%), 0.4 (40%), 0.6(60%), 0.8(80%) and 1 (100%) for the exercise 1.3 and 0.01 (1%), 0.02 (2%), 0.05(5%) and 0.1 (10%) for the exercise 1.4. Each folder has the log file with the output

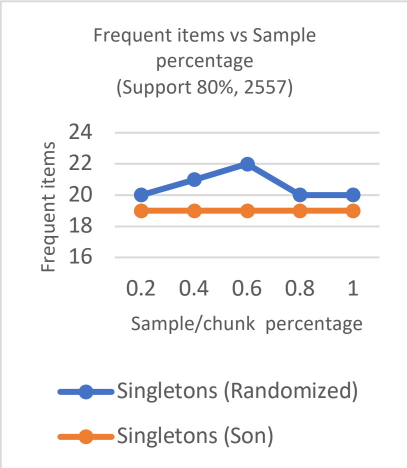
1.3.1 Comparison on chess dataset

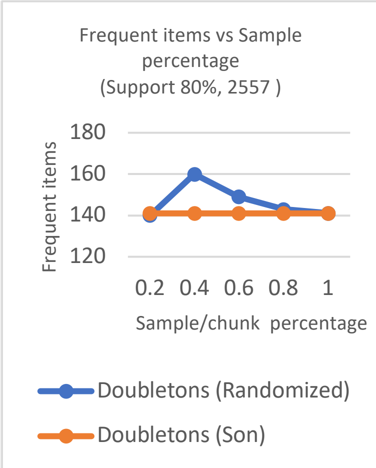
Frequent items

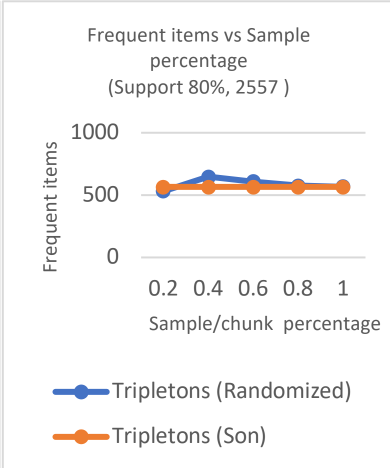
In the following table are described all the experiments performed with Randomized and Son algorithm on chess dataset. For different supports {20%,40,60%,80%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: Chess Dataset size : 3196																				
Support	20%	20%	20%	20%	20%	40%	40%	40%	40%	40%	60%	60%	60%	60%	60%	80%	80%	80%	80%	80%
Support count (threshold)	639	639	639	639	639	1278	1278	1278	1278	1278	1918	1918	1918	1918	1918	2557	2557	2557	2557	2557
Effective threshold (support count)	128	256	384	511	639	256	511	767	1023	1278	384	767	1151	1534	1918	511	1023	1534	2045	2557
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	53	53	53	54	54	44	40	40	40	40	33	34	34	34	34	20	21	22	20	20
Singletons (Son)	54	54	54	54	54	40	40	40	40	40	34	34	34	34	34	19	19	19	19	19
Doubletons (Randomized)	1160	1161	1156	1168	1161	711	671	671	674	673	369	385	384	412	389	140	160	149	143	141
Doubletons (Son)	1161	1161	1161	1161	1161	673	673	673	673	673	389	389	389	389	389	141	141	141	141	141
Tripletons (Randomized)	14098	13980	13870	14013	13850	6607	6098	6126	6191	6117	2205	2350	2271	2529	2325	531	648	608	575	568
Tripletons (Son)	13850	13850	13850	13850	13850	6117	6117	6117	6117	6117	2325	2325	2325	2325	2325	566	566	566	566	566
Runtime (Randomized) (milliseconds)	1513	2399	3788	5646	6625	1285	2001	3181	4659	5114	997	1458	2968	3468	3499	599	929	2237	1738	2529
Runtime (Son) (milliseconds)	11256	11795	11526	12440	9758	7759	7468	8274	7118	7433	6918	7186	7152	6681	4984	3594	4025	4033	2918	3399
Memory used (Randomized) (KB)	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	12632	13213	13374	13338	13247	13241
Memory used (Son) (KB)	12641	12641	12642	12642	12642	12641	12641	12642	12642	12642	12642	18746	18938	18070	16874	13980	13980	13981	13977	13698

In the appendix 4.1 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%







Observations:

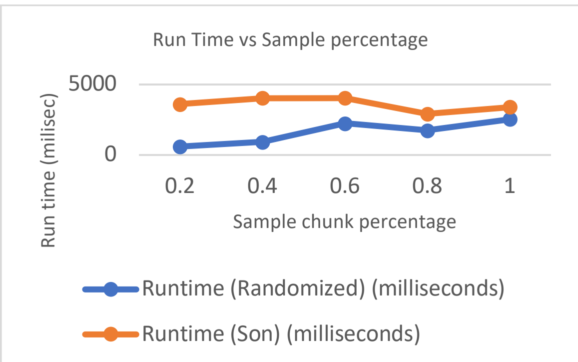
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

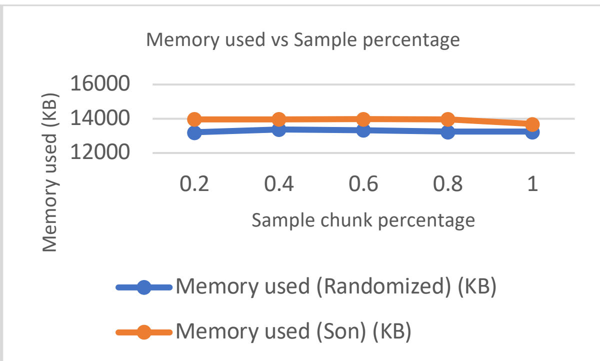
When the sample/chunk is equal to the 100% or when the sample is large enough, in the most of the cases, both algorithm has the same results in term of frequent item set such as singleton, pairs and tripletons, which make sense because with sample size 100%, randomized is a-priori algorithm

When the sample/chunk are approaching to 0, in the most of the cases, the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms, which suggests the randomized algorithm identify false positives and false negatives.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions:
According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item, although due to simplicity lower sample sizes will be explored in the next section. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

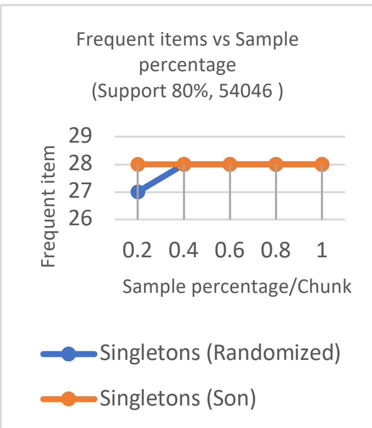
1.3.2 Comparison on connect dataset

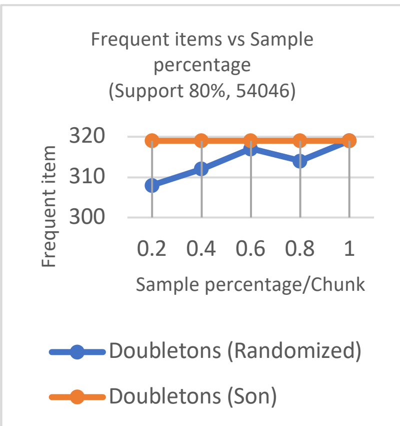
Frequent items

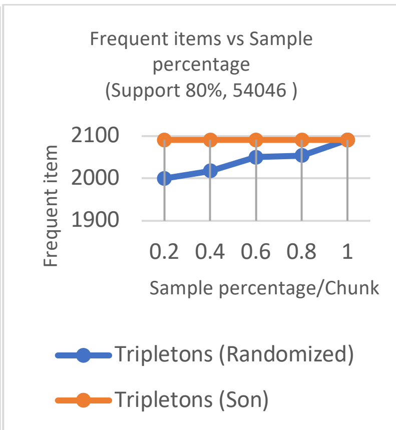
In the following table are described all the experiments performed with Randomized and Son algorithm on connect dataset. For different supports {20%,40,60%,80%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: Connect Dataset size : 67557																				
Support	20%	20%	20%	20%	20%	40%	40%	40%	40%	40%	60%	60%	60%	60%	60%	80%	80%	80%	80%	80%
Support count (threshold)	13511	13511	13511	13511	13511	27023	27023	27023	27023	27023	40534	40534	40534	40534	40534	54046	54046	54046	54046	54046
Effective threshold (support count)	2702.3	5404.6	8106.8	10809	13511	5404.6	10809	16214	21618	27023	8106.8	16214	24321	32427	40534	10809	21618	32427	43236	54046
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	59	59	59	59	59	41	41	41	41	41	36	36	36	36	36	27	28	28	28	28
Singletons (Son)	59	59	59	59	59	41	41	41	41	41	36	36	36	36	36	28	28	28	28	28
Doubletons (Randomized)	1364	1358	1361	1357	1358	751	747	744	745	744	542	533	533	538	539	308	312	317	314	319
Doubletons (Son)	1358	1358	1358	1358	1358	744	744	744	744	744	539	539	539	539	539	319	319	319	319	319
Tripletons (Randomized)	17871	17744	17781	17709	17738	8154	8101	8018	8051	8039	4782	4677	4681	4718	4737	2000	2018	2050	2054	2091
Tripletons (Son)	17738	17738	17738	17738	17738	8039	8039	8039	8039	8039	4737	4737	4737	4737	4737	2091	2091	2091	2091	2091
Runtime (Randomized) (milliseconds)	32765	75739	85338	118487	176163	20593	41010	55359	74681	97252	16754	29726	51508	56569	108013	10451	22226	31063	39000	45374
Runtime (Son) (milliseconds)	353069	349314	328880	320392	309773	197099	194270	191333	187298	161679	133907	154628	176792	167371	150780	93075	96981	106261	107949	110867
Memory used (Randomized) (KB)	12634	12634	12634	12634	12634	12611	12611	12634	12634	12634	12634	12634	12634	12634	12634	12634	12634	12634	12634	12634
Memory used (Son) (KB)	12646	12646	12647	12647	12647	12646	12646	12647	12647	12647	12646	12646	12647	12647	12647	16784	17062	17080	17025	16219

In the appendix 4.2 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%







Observations:
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

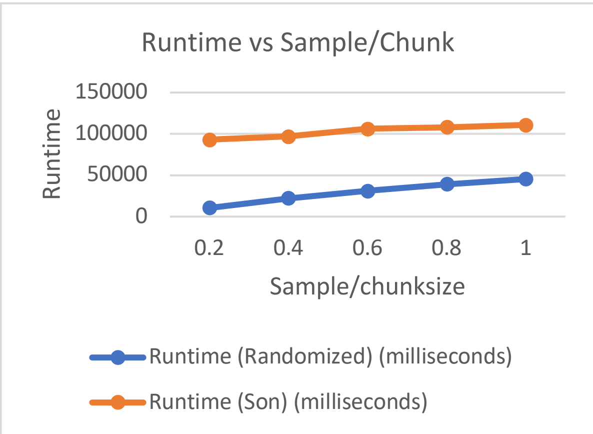
When the sample/chunk is equal to the 100% or when the sample is large enough, in the most of the cases, both algorithm has the same results in term of frequent item set such as singleton, pairs and tripletons, which make sense because with sample size 100%, randomized is a-priori algorithm

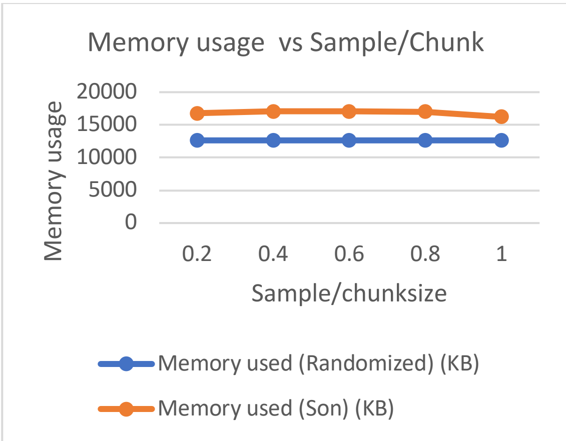
When the sample/chunk are approaching to 0, in the most of the cases, the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms, which suggests the randomized algorithm identify false positives and false negatives.

In general, in this dataset, for support above than 50%, there are more false negatives that Randomized algorithm identified and for support below 50%, there are more false positives.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions:

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item, although due to simplicity lower sample sizes will be explored in the next section. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

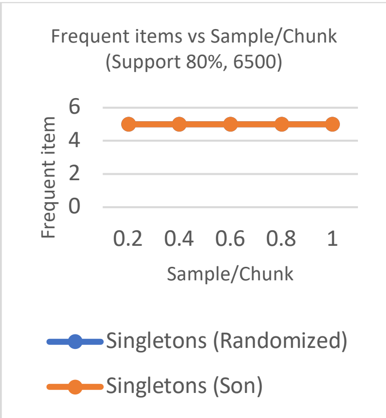
1.3.3 Comparison on mushroom dataset

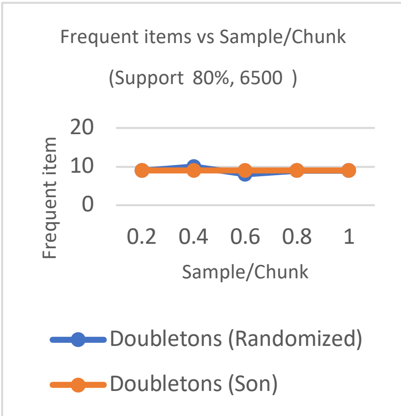
Frequent items

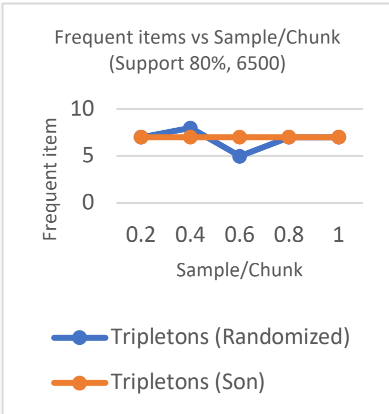
In the following table are described all the experiments performed with Randomized and Son algorithm on mushroom dataset. For different supports {20%,40,60%,80%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: Mushroom Dataset size : 8124																				
Support	20%	20%	20%	20%	20%	40%	40%	40%	40%	40%	60%	60%	60%	60%	60%	80%	80%	80%	80%	80%
Support count (threshold)	1625	1625	1625	1625	1625	3250	3250	3250	3250	3250	4874	4874	4874	4874	4874	6500	6500	6500	6500	6500
Effective threshold (support count)	325	650	975	1300	1625	650	1300	1950	2600	3250	974.8	1950	2924	3899	4874	1300	2600	3900	5200	6500
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	42	42	43	43	43	22	21	22	21	21	8	8	7	8	8	5	5	5	5	5
Singletons (Son)	43	43	43	43	43	21	21	21	21	21	8	8	8	8	8	5	5	5	5	5
Doubletons (Randomized)	354	372	374	378	376	105	98	99	97	97	18	18	16	18	18	9	10	8	9	9
Doubletons (Son)	376	376	376	376	376	97	97	97	97	97	18	18	18	18	18	9	9	9	9	9
Tripletons (Randomized)	1311	1463	1457	1486	1472	205	198	187	181	185	17	17	16	19	17	7	8	5	7	7
Tripletons (Son)	1472	1472	1472	1472	1472	185	185	185	185	185	17	17	17	17	17	7	7	7	7	7
Runtime (Randomized) (milliseconds)	1122	2593	3191	3618	3427	708	1232	1565	1633	2253	434	790	938	1059	1442	407	694	943	1130	1189
Runtime (Son) (milliseconds)	6197	6144	6685	5818	5611	3796	3763	4207	4156	3383	2247	2313	2147	1970	2219	1970	1976	1860	1845	1717
Memory used (Randomized) (KB)	12633	12633	12633	12633	12633	12633	12633	12633	12633	12633	12633	12633	12645	12645	12645	12651	12633	12645	12645	12644
Memory used (Son) (KB)	18150	17993	17282	16679	16189	13664	13520	13484	13312	13186	12641	12641	12643	12734	12643	12644	12644	12645	12645	12645

In the appendix 4.3 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%







<u>Observations:</u>

As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

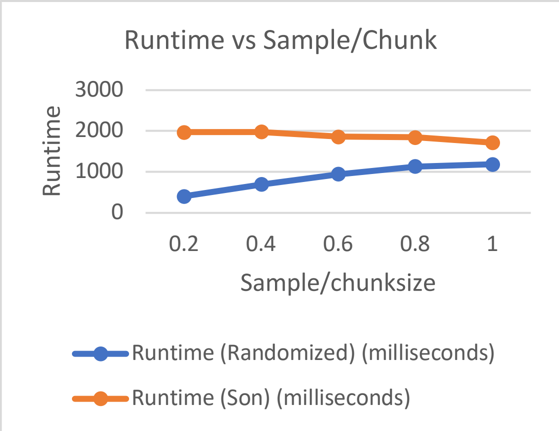
When the sample/chunk is equal to the 100% or when the sample is large enough, in the most of the cases, both algorithm has the same results in term of frequent item set such as singleton, pairs and tripletons, which make sense because with sample size 100%, randomized is a-priori algorithm

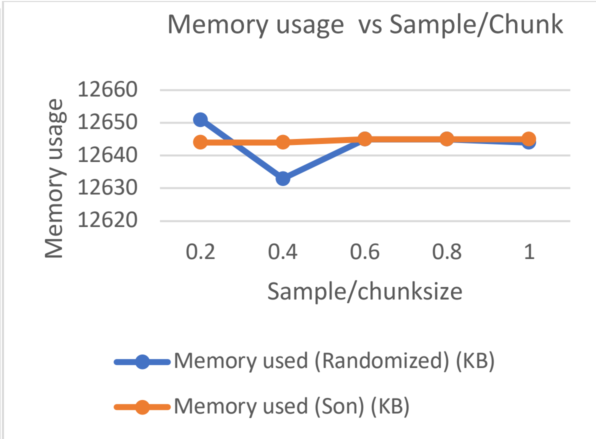
When the sample/chunk are approaching to 0, in the most of the cases, the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms, which suggests the randomized algorithm identify false positives and false negatives.

In general, in this dataset, for the largest support, there are less false negatives and false positives by the implementation of Randomized algorithm.

<u>Runtime and memory usage</u>

The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes. Randomized algorithm outperform Son Algorithm in terms of run time, however in terms of memory usage randomized algorithm only outperform Son algorithm for a sample size of 40%.





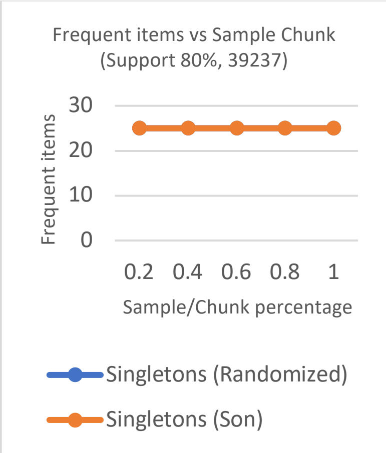
<u>Decisions:</u>
According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item, although due to simplicity lower sample sizes will be explored in the next section. Even though, the randomized algorithm is much more efficient in term of run time, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

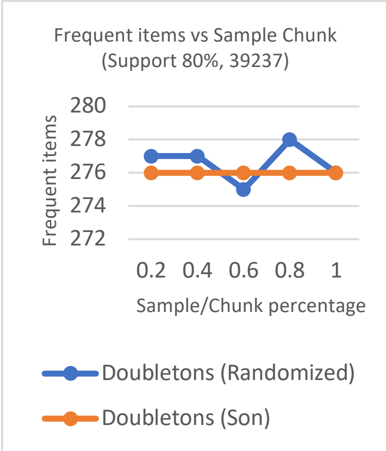
1.3.4 Comparison on pumsb dataset

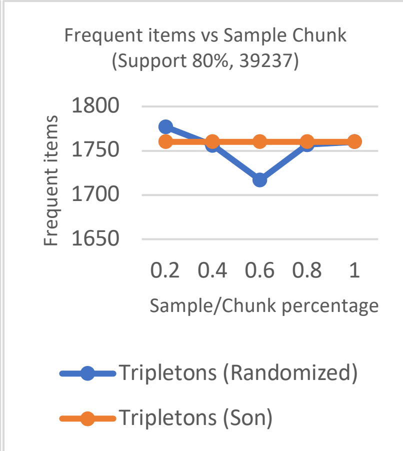
<u>Frequent items</u>
In the following table are described all the experiments performed with Randomized and Son algorithm on pumsb dataset. For different supports {20%,40,60%,80%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: Pumb Dataset size : 49046																				
Support	20%	20%	20%	20%	20%	40%	40%	40%	40%	40%	60%	60%	60%	60%	60%	80%	80%	80%	80%	80%
Support count (threshold)	1883	1883	1883	1883	1883	9417	9417	9417	9417	9417	23542	23542	23542	23542	23542	39237	39237	39237	39237	39237
ffective threshold (support count	377	753	1130	1506	1883	1883	3767	5650	7533	9417	4708	9417	14125	18834	23542	7847	15695	23542	31389	39237
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	194	194	191	191	190	112	112	112	112	112	57	59	59	59	59	25	25	25	25	25
Singletons (Son)	190	190	190	190	190	112	112	112	112	112	59	59	59	59	59	25	25	25	25	25
Doubletons (Randomized)	10047	10074	10019	9975	9966	4237	4235	4226	4220	4226	1046	1110	1107	1116	1108	277	277	275	278	276
Doubletons (Son)	9966	9966	9966	9966	9966	4226	4226	4226	4226	4226	1108	1108	1108	1108	1108	276	276	276	276	276
Tripletons (Randomized)	274490	276224	275064	274462	274142	85768	85211	85042	84636	84805	10703	11482	11410	11589	11368	1777	1756	1717	1757	1760
Tripletons (Son)	274142	274142	274142	274142	274142	84805	84805	84805	84805	84805	11368	11368	11368	11368	11368	1760	1760	1760	1760	1760
Runtime (Randomized) (millisec)	256043	548059	816498	1096689	1173736	119539	233743	369162	475118	591801	44111	89917	131516	178247	218378	7847	27935	42248	64965	83741
Runtime (Son) (milliseconds)	2585227	2520698	2338672	2357535	2346909	1438204	1472383	136438	1453119	1435549	517750	559898	616574	592433	567417	194226	187073	158169	169732	174705
Memory used (Randomized) (KB)	292141	293400	292500	292056	291833	9907648	9869736	98569	98272	98398	12634	12636	12636	12636	12636	12634	14397	14367	12636	14398
Memory used (Son) (KB)	291798	291798	291799	291799	291799	98406	98406	98407	98409	98407	12643	12643	12647	1264464	12644	15830	16047	16049	16048	15733

In the appendix 4.4 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%







Observations:

As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

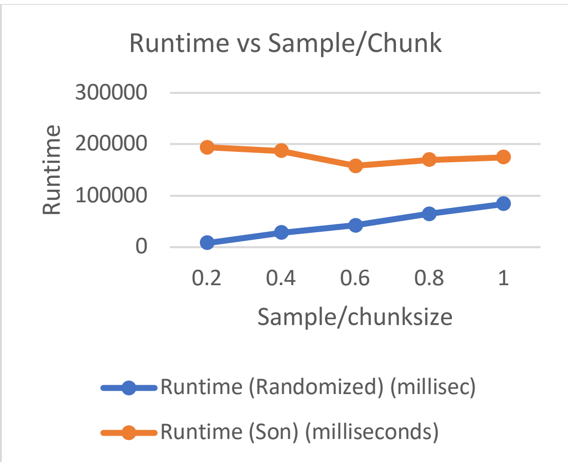
When the sample/chunk is equal to the 100% or when the sample is large enough, in the most of the cases, both algorithm has the same results in term of frequent item set such as singleton, pairs and tripletons, which make sense because with sample size 100%, randomized is a-priori algorithm

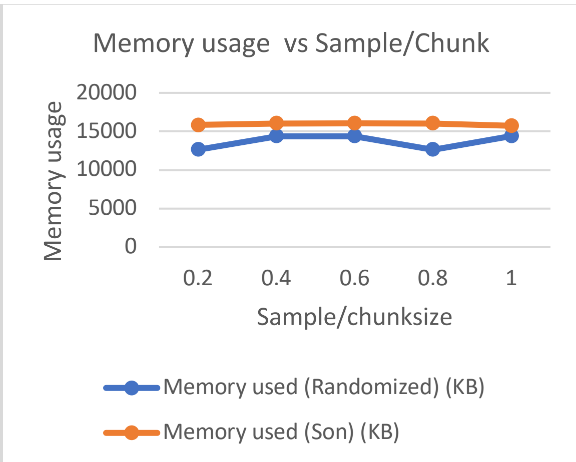
When the sample/chunk are approaching to 0, in the most of the cases, the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms, which suggests the randomized algorithm identify false positives and false negatives.

In general, in this dataset, for the largest support, there are less false negatives and false positives by the implementation of Randomized algorithm.

Runtime and memory usage

The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes. Randomized algorithm outperform Son Algorithm in terms of run time and memory usage.





Decisions:

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item, although due to simplicity lower sample sizes will be explored in the next section. Even though, the randomized algorithm is much more efficient in term of run time, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

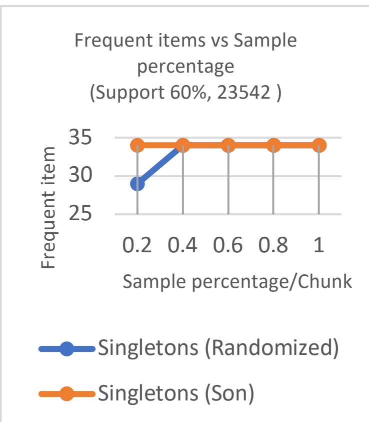
1.3.5 Comparison on pumsb star dataset

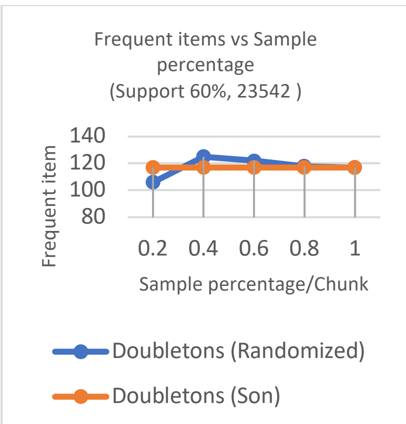
Frequent items

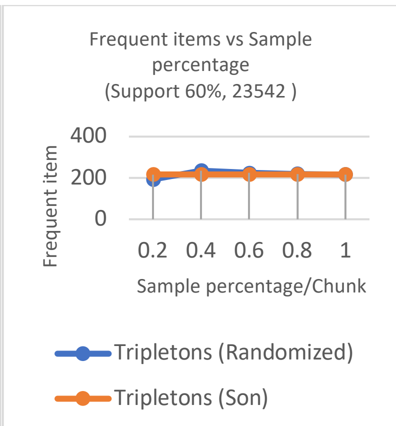
In the following table are described all the experiments performed with Randomized and Son algorithm on pumsb star dataset. For different supports {20%,40,60%,80%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: Pumb_star Dataset size : 49046																				
Support	20%	20%	20%	20%	20%	40%	40%	40%	40%	40%	60%	60%	60%	60%	60%	80%	80%	80%	80%	80%
Support count (threshold)	1883	1883	1883	1883	1883	9417	9417	9417	9417	9417	23542	23542	23542	23542	23542	39237	39237	39237	39237	39237
Effective threshold (support count)	377	753	1130	1506	1883	1883	3767	5650	7533	9417	4708	9417	14125	18834	23542	7847	15695	23542	31389	39237
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	167	168	164	165	165	87	87	88	87	87	29	34	34	34	34	0	0	0	0	0
Singletons (Son)	165	165	165	165	165	87	87	87	87	87	34	34	34	34	34	0	0	0	0	0
Doubletons (Randomized)	5690	5733	5669	5657	5667	1816	1804	1799	1806	1808	106	125	122	118	117	0	0	0	0	0
Doubletons (Son)	5667	5667	5667	5667	5667	1808	1808	1808	1808	1808	117	117	117	117	117	0	0	0	0	0
Tripletons (Randomized)	90327	90852	89798	89755	89731	16258	16148	16162	16265	16281	193	235	224	219	217	0	0	0	0	0
Tripletons (Son)	89731	89731	89731	89731	89731	16281	16281	16281	16281	16281	217	217	217	217	217	0	0	0	0	0
Runtime (Randomized) (milliseconds)	72507	140018	199749	293604	366103	33509	71594	112170	147997	181607	7569	19571	24724	33077	55589	5059	7541	10427	13998	15947
Runtime (Son) (milliseconds)	676413	699816	739137	646577	612036	312400	315534	365319	331040	294024	60486	63993	69744	66691	62073	30712	30848	34065	36585	35226
Memory used (Randomized) (KB)	124921	125330	124523	124484	124476	12634	12634	12637	12637	12637	13274	13563	13556	13550	13547	11782	11784	11784	11784	11784
Memory used (Son) (KB)	124484	124484	124485	124485	124485	12644	12644	12647	12644	12644	13880	14084	14736	14048	13775	11792	11792	11793	11795	11793

In the appendix 4.5 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 60%.







Observations:

As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

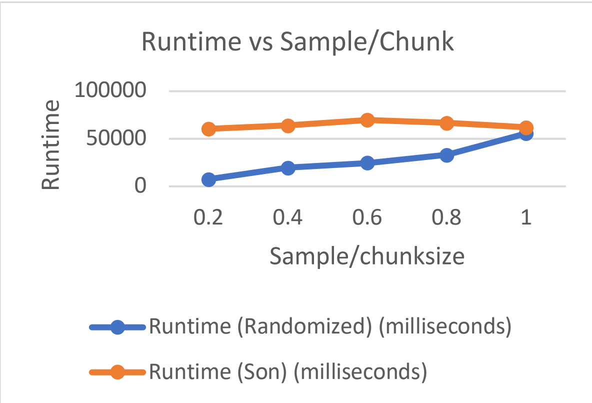
When the sample/chunk is equal to the 100% or when the sample is large enough, in the most of the cases, both algorithm has the same results in term of frequent item set such as singleton, pairs and tripletons, which make sense because with sample size 100%, randomized is a-priori algorithm

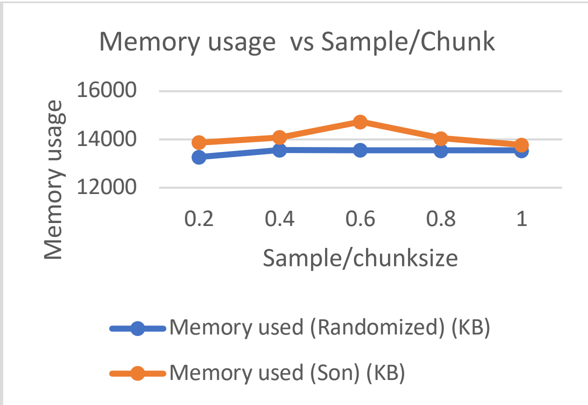
When the sample/chunk are approaching to 0, in the most of the cases, the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms, which suggests the randomized algorithm identify false positives and false negatives.

In general, in this dataset, for the largest support and larger samples, there are less false negatives and false positives by the implementation of Randomized algorithm.

Runtime and memory usage

The following plots show the runtime and memory usage for both algorithms for support threshold 60% and its different sample/chunk sizes. Randomized algorithm outperform Son Algorithm in terms of run time and memory usage.





Decisions:

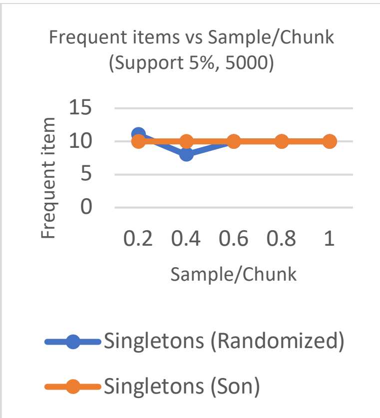
According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 60% because we can obtain a manageable number of frequent item, although due to simplicity lower sample sizes will be explored in the next section. Even though, the randomized algorithm is much more efficient in term of run time, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

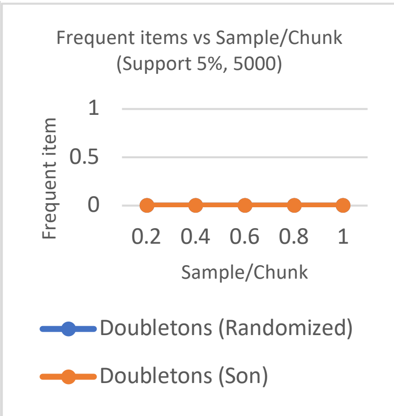
1.3.6 Comparison on T10I4D100K dataset

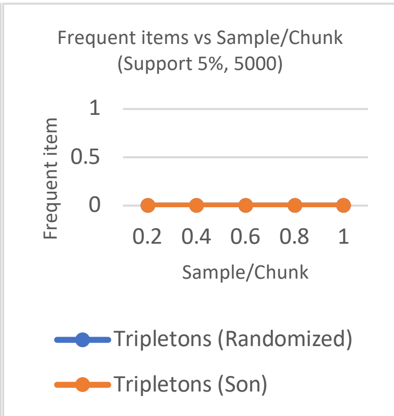
In the following table are described all the experiments performed with Randomized and Son algorithm on T10I4D100K dataset. In this dataset, the support threshold was decreased, because for a support more than 20%, no frequent items appear. For different supports {5%,10%,15%,20%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset: T10I4D100K Dataset size : 100000																				
Support	5%	5%	5%	5%	5%	10%	10%	10%	10%	10%	15%	15%	15%	15%	15%	20%	20%	20%	20%	20%
Support count (threshold)	5000	5000	5000	5000	5000	10000	10000	10000	10000	10000	15000	15000	15000	15000	15000	20000	20000	20000	20000	20000
Effective threshold (support count)	1000	2000	3000	4000	5000	2000	4000	6000	8000	10000	3000	6000	9000	12000	15000	4000	8000	12000	16000	20000
Sample	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	11	8	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Singletons (Son)	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Runtime (Randomized) (millisecs)	1068	1360	1777	1807	1967	978	1440	1468	1846	1563	1058	1369	1713	1672	1842	900	1428	1631	1665	1593
Runtime (Son) (milliseconds)	3015	2783	2623	3318	2643	2403	2699	3052	2432	2561	2612	2550	2357	2539	2396	2612	2550	2436	2525	2441
Memory used (Randomized) (KB)	11661	11661	11744	11744	11744	11654	11654	11654	11721	11721	11654	11654	11721	11721	11721	11654	11654	11721	11721	11721
Memory used (Son) (KB)	11755	11755	11756	11756	11756	11730	11730	11731	11731	11731	11730	11730	11731	11731	11731	11730	11730	11731	11731	11731

In the appendix 4.6 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 5%.







Observations:

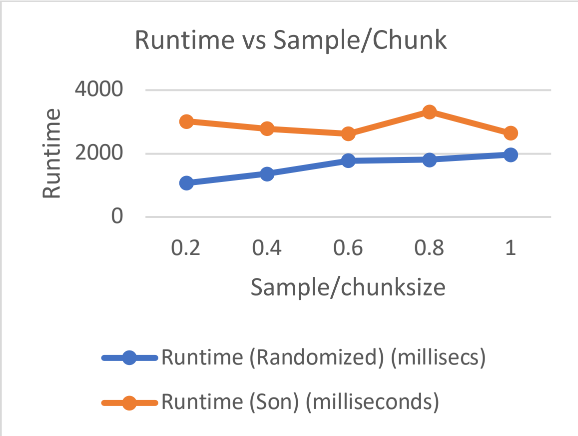
We got only frequent items for a 5% support, which suggest we need to explore lower support threshold

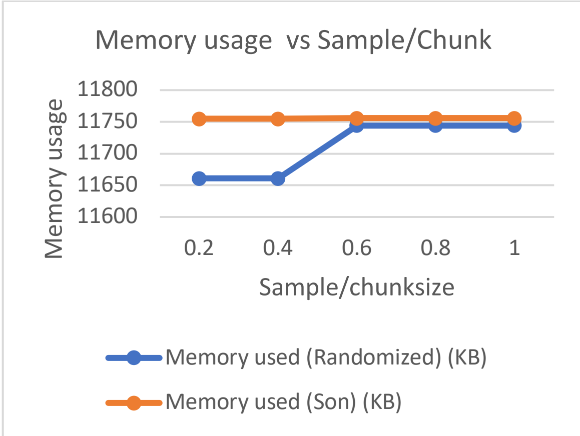
For the single output with results, when the sample/chunk is approaching to the 100%, both algorithm has the same results in term of frequent item.

For the single output, when the sample/chunk are approaching to 0, , the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms.

Runtime and memory usage

The following plots show the runtime and memory usage for both algorithms for support threshold 5% and its different sample/chunk sizes. Randomized algorithm outperform Son Algorithm in terms of run time and memory usage.





Decisions:

According to the data provided, assuming that we don’t know additional information about association rules or the problem that need to be solved or answered, the support should be selected is lower than 5% because we can obtain frequent item, lower sample sizes will be explored in the next section.

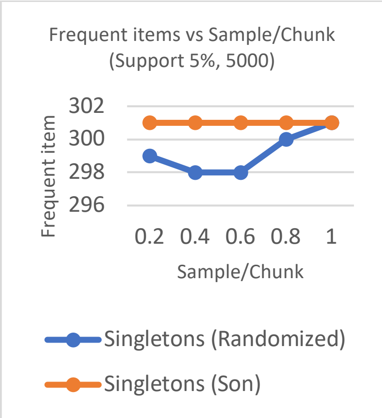
1.3.7 Comparison on T40I10D100K dataset

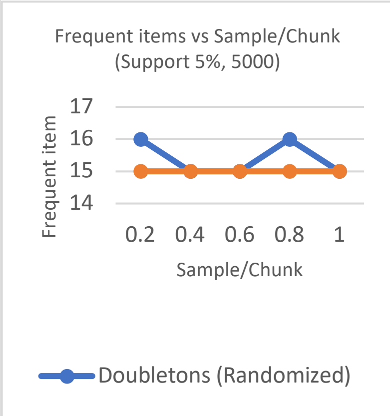
Frequent itemset

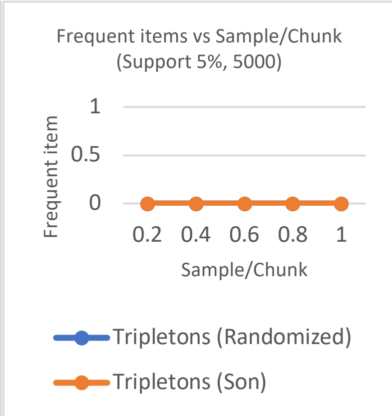
In the following table are described all the experiments performed with Randomized and Son algorithm on T40I10D100K dataset. In this dataset, the support threshold was decreased, because for a support more than 20%, frequent items appear in lower quantities. For different supports {5%,10%,15%,20%} were experimented different sample/chunk size {0.2, 0.4, 0.6, 0.8, 1} and were obtained as results singletons size, pairs size, tripletons size, memory usage and run time.

Dataset:T40I10D100K Dataset size : 100000																				
Support	5%	5%	5%	5%	5%	10%	10%	10%	10%	10%	15%	15%	15%	15%	15%	20%	20%	20%	20%	20%
Support count (threshold)	5000	5000	5000	5000	5000	10000	10000	10000	10000	10000	15000	15000	15000	15000	15000	20000	20000	20000	20000	20000
Effective threshold (support count)	1000	2000	3000	4000	5000	2000	4000	6000	8000	10000	3000	6000	9000	12000	15000	4000	8000	12000	16000	20000
Sample/Chunk	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
Singletons (Randomized)	299	298	298	300	301	80	83	81	82	82	18	19	19	20	19	5	4	5	5	5
Singletons (Son)	301	301	301	301	301	82	82	82	82	82	19	19	19	19	19	5	5	5	5	5
Doubletons (Randomized)	16	15	15	16	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Son)	15	15	15	15	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Runtime (Randomized) (millisecs)	76304	144724	232744	340153	433220	8651	15323	23087	31214	42488	4421	8720	11992	15883	18638	4925	7806	13954	16408	19223
Runtime (Son) (millisecs)	372651	414619	457021	456891	446096	53800	51482	58324	51774	51812	32592	35803	41969	39221	35552	38074	43135	37562	33056	32726
Memory used (Randomized) (KB)	278531	278532	278544	278534	278534	22509	22510	22509	22509	22509	11817	11818	11818	11818	11817	11734	11732	11734	11734	11734
Memory used (Son) (KB)	278554	278586	278736	278584	278553	22521	22519	22526	22520	22522	11828	11828	11829	11827	11827	11744	11744	11743	11743	11745

In the appendix 4.7 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 5%.





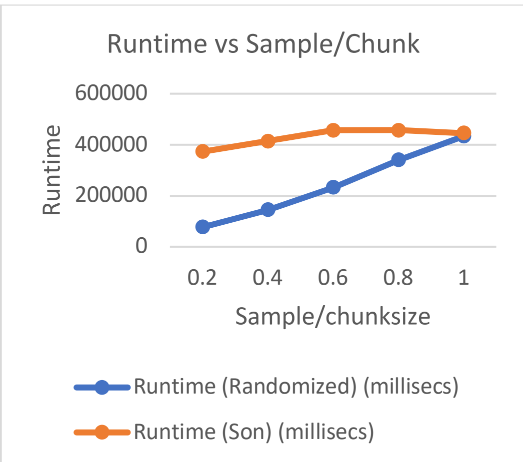


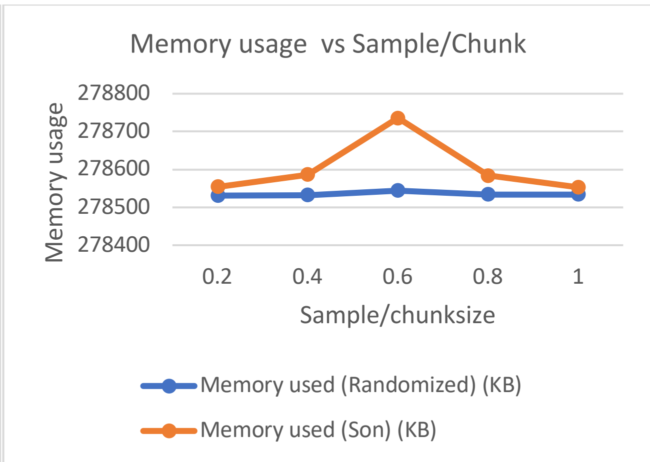
Observations:
We got only singletons and doubletons for a 5% support, for support more than 5%, we only obtain singletons

As a rule of thumb, when the sample/chunk is approaching to the 100%, both algorithm has the same results in term of frequent item.

When the sample/chunk are approaching to 0, , the variation of the frequent items set between the randomized algorithm and son algorithm is going up in absolute terms.

Runtime and memory usage
The following plots show the runtime and memory usage for both algorithms for support threshold 5% and its different sample/chunk sizes. Randomized algorithm outperform Son Algorithm in terms of run time and memory usage. For sample equal to the dataset size, run time and memory usage converge, which make sense.





Decisions:
According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is lower than 5% because we can obtain manageable singletons and doubletons, lower sample sizes will be explored in the next section.

1.3.8 General Discussion

Randomized algorithm has huge variation in term of singletons, doubletons, tripletons, which suggest that the algorithm identify false negatives and false positives.

The implementation of Randomized algorithm is more efficient in term of memory usage and runtime, since use a sample of the dataset and use the same sample for both passes.

Lower sample sizes, more false positives and false negatives for the implementation of Randomized algorithm

1.4 Experiment with different sample sizes

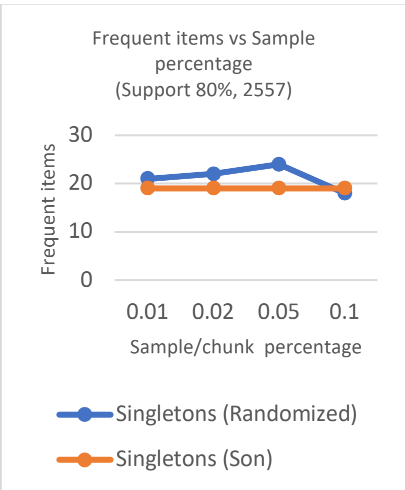
1.4.1 Comparison on chess dataset

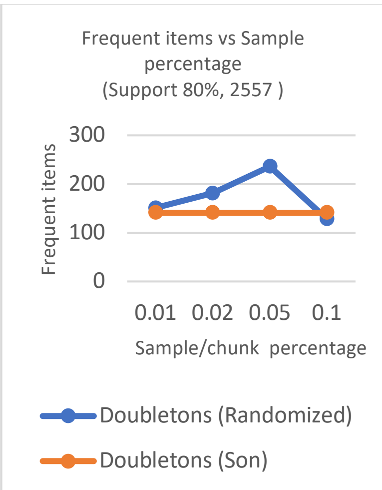
Frequent items

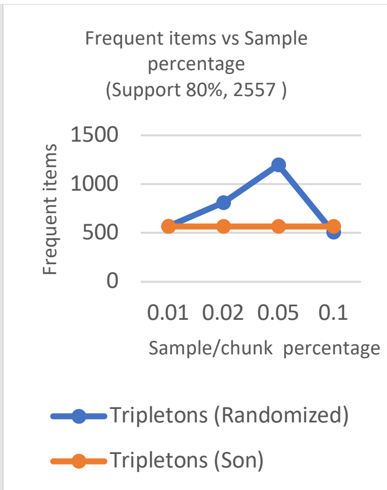
In the following table are described all the experiments performed with Randomized and Son algorithm on chess dataset. For different supports {20%,40,60%,80%} were experimented lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: Chess Dataset size : 3196																
Support	20%	20%	20%	20%	40%	40%	40%	40%	60%	60%	60%	60%	80%	80%	80%	80%
Support count (threshold)	639	639	639	639	1278	1278	1278	1278	1918	1918	1918	1918	2557	2557	2557	2557
Effective threshold (support count)	6	13	32	64	13	26	64	128	19	38	96	192	26	51	128	256
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	54	53	54	54	43	43	42	40	30	36	29	31	21	22	24	18
Singletons (Son)	54	54	54	54	40	40	40	40	34	34	34	34	19	19	19	19
Doubletons (Randomized)	1131	1098	1188	1161	767	760	688	667	353	491	296	355	151	181	236	128
Doubletons (Son)	1161	1161	1161	1161	673	673	673	673	389	389	389	389	141	141	141	141
Tripletons (Randomized)	13467	12706	14636	13844	7606	7841	6290	6116	2296	3442	1611	2176	566	809	1197	502
Tripletons (Son)	13850	13850	13850	13850	6117	6117	6117	6117	2325	2325	2325	2325	566	566	566	566
Runtime (Randomized) (milliseconds)	721	786	921	1174	570	633	748	906	346	481	494	607	250	301	448	445
Runtime (Son) (milliseconds)	12170	12466	11733	10689	8834	7756	7472	7500	6606	6076	5692	6546	5182	4106	4060	3146
Memory used (Randomized) (KB)	12652	12610	12634	12634	12652	12610	12634	12634	12652	12652	14071	12634	12650	12652	12652	12652
Memory used (Son) (KB)	12620	12643	12463	12643	12643	12643	12643	12643	12643	12643	12643	18948	13982	13982	13982	13982

In the appendix 4.8 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%.







Observations:

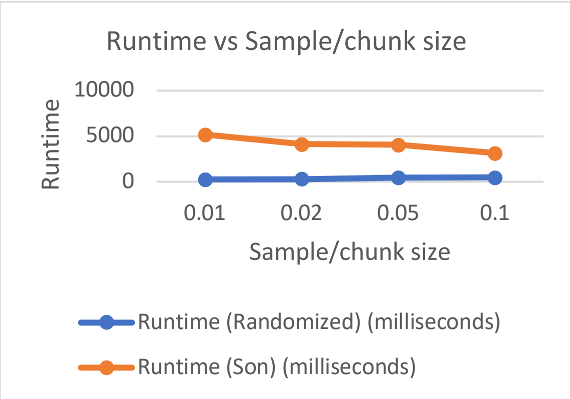
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

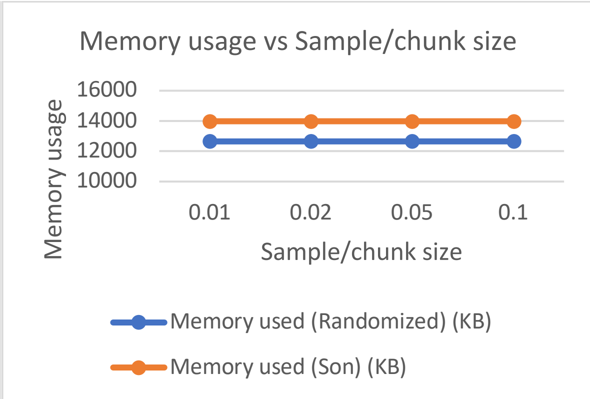
If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise

In general, one important difference with respect to the previous exercise, for large support (60%,80%) and small support (20%), when the sample size is small enough (0.01 or 0.02), the variation in frequent items for both algorithms is decreasing when the sample size is approaching to 0, which suggest, we can use the randomized algorithm for this support and sample sizes..

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

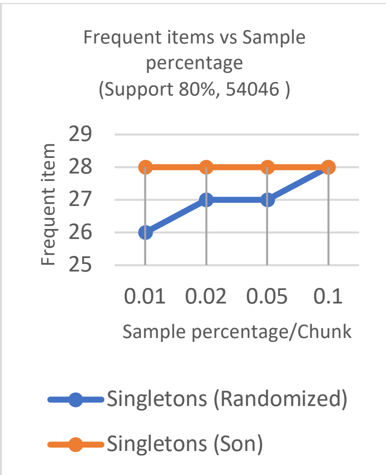
1.4.2 Comparison on connect dataset

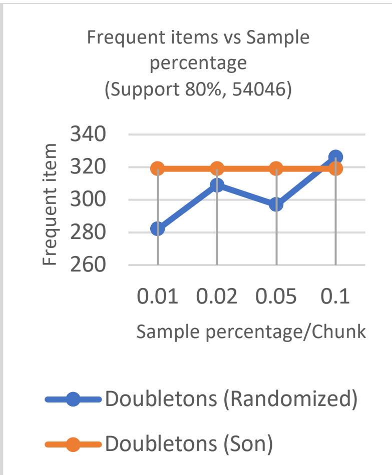
Frequent items

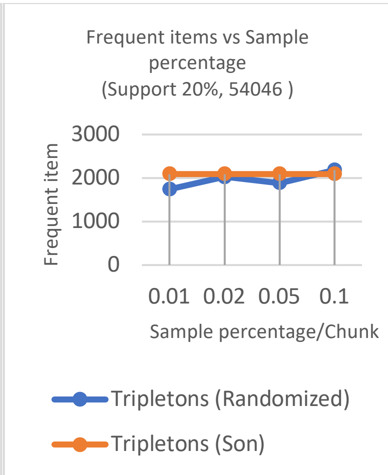
In the following table are described all the experiments performed with Randomized and Son algorithm on connect dataset. For different supports {20%,40,60%,80%} were experimented lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: Connect Dataset size : 67557																
Support	20%	20%	20%	20%	40%	40%	40%	40%	60%	60%	60%	60%	80%	80%	80%	80%
Support count (threshold)	13511	13511	13511	13511	27023	27023	27023	27023	40534	40534	40534	40534	54046	54046	54046	54046
ffective threshold (support count	135	270	676	1351	270	540	1351	2702	405	811	2027	4053	540	1081	2702	5405
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	61	61	59	60	40	41	41	41	36	36	36	35	26	27	27	28
Singletons (Son)	59	59	59	59	41	41	41	41	36	36	36	36	28	28	28	28
Doubletons (Randomized)	1375	1400	1347	1392	710	732	747	744	564	550	544	537	282	309	297	326
Doubletons (Son)	1358	1358	1358	1358	744	744	744	744	539	539	539	539	319	319	319	319
Tripletons (Randomized)	17749	18252	17435	18234	7571	7907	8114	8041	5121	4925	4853	4862	1742	2028	1884	2184
Tripletons (Son)	17738	17738	17738	17738	8039	8039	8039	8039	4737	4737	4737	4737	2091	2091	2091	2091
Runtime (Randomized) (millisecs	2517	3985	7754	14117	1879	2540	5462	8560	1649	3622	4808	7684	1119	1921	3071	5424
Runtime (Son) (milliseconds)	279305	279871	298789	306158	162904	153729	147946	149428	139481	144799	154271	15180	92291	103739	103436	95303
Memory used (Randomized) (KB)	12635	12635	12635	12635	12635	12635	12635	12635	12635	12635	12635	12635	14405	12635	12635	12635
Memory used (Son) (KB)	12644	12644	12646	12646	12646	12644	12644	12646	12646	12646	12646	12644	17079	17080	17065	17023

In the appendix 4.9 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%.







Observations:

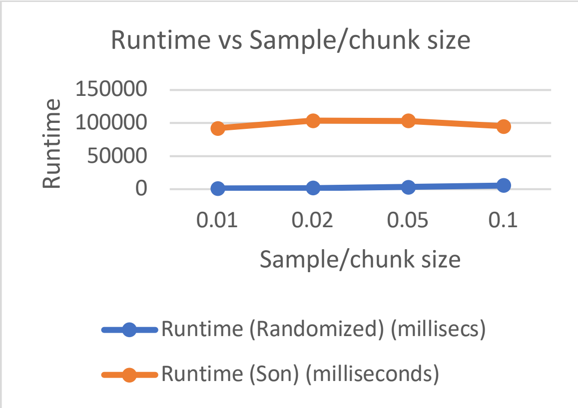
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

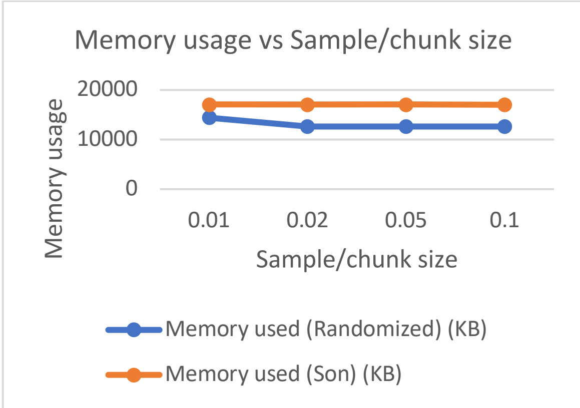
If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise

In general, if the sample is small enough, in general the randomized algorithm produce more false negatives than previous exercise with larger samples sizes.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

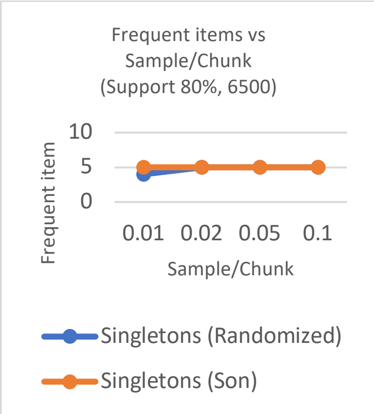
1.4.3 Comparison on mushroom dataset

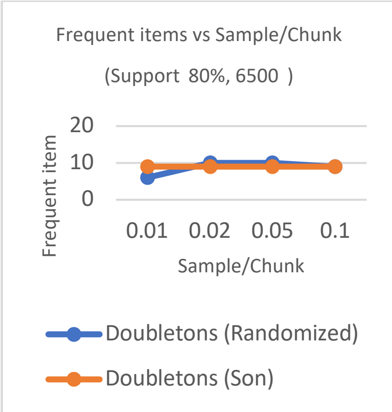
Frequent items

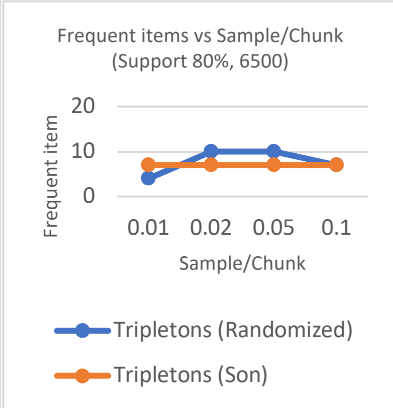
In the following table are described all the experiments performed with Randomized and Son algorithm on mushroom dataset. For different supports {20%,40,60%,80%} were experimented lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: Mushroom Dataset size : 8124																
Support	20%	20%	20%	20%	40%	40%	40%	40%	60%	60%	60%	60%	80%	80%	80%	80%
Support count (threshold)	1625	1625	1625	1625	3250	3250	3250	3250	4874	4874	4874	4874	6500	6500	6500	6500
Effective threshold (support count)	16	33	81	163	33	65	163	325	49	97	244	487	65	130	325	650
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	45	41	43	44	16	17	24	21	15	12	7	9	4	5	5	5
Singletons (Son)	43	43	43	43	21	21	21	21	8	8	8	8	5	5	5	5
Doubletons (Randomized)	455	379	369	387	68	67	132	98	58	40	14	20	6	10	10	9
Doubletons (Son)	376	376	376	376	97	97	97	97	18	18	18	18	9	9	9	9
Tripletons (Randomized)	1997	1513	1374	1527	130	110	305	187	91	54	13	20	4	10	10	7
Tripletons (Son)	1472	1472	1472	1472	185	185	185	185	17	17	17	17	7	7	7	7
Runtime (Randomized) (milliseconds)	374	427	526	666	226	251	367	425	231	234	273	335	201	211	276	305
Runtime (Son) (milliseconds)	6296	6601	6477	5807	3132	3085	2922	3133	1834	2122	1883	1902	1826	1821	1855	1872
Memory used (Randomized) (KB)	12652	12652	12634	12634	12650	12652	12652	12634	12650	12652	12652	12652	12650	12650	12652	12652
Memory used (Son) (KB)	12643	19870	19556	19161	13852	13825	13803	13975	12746	12746	12746	12747	12688	12688	12688	12688

In the appendix 4.10there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%.







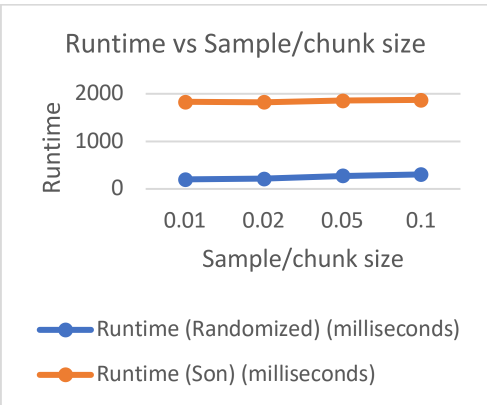
Observations:

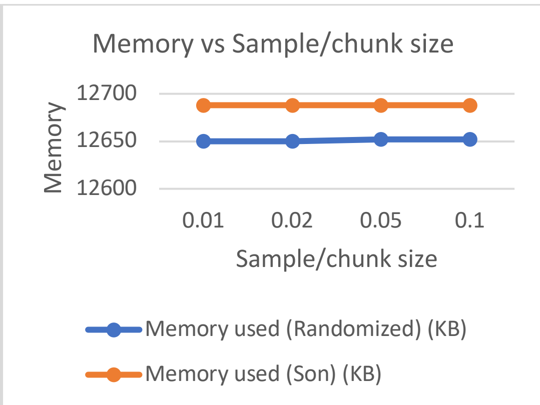
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

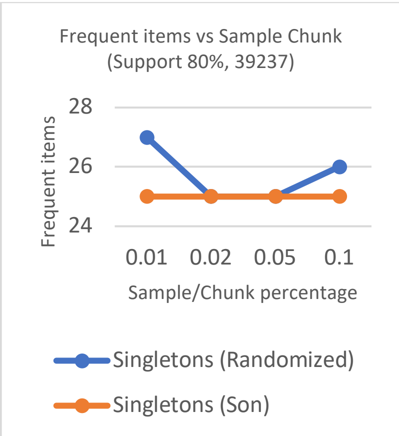
1.4.4 Comparison on pumsb dataset

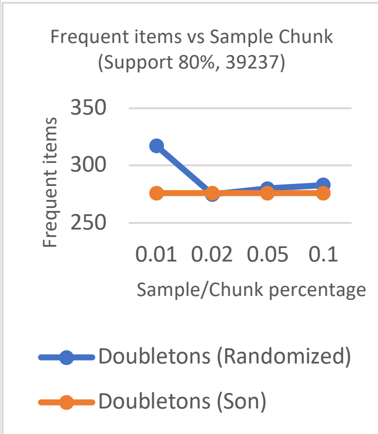
Frequent items

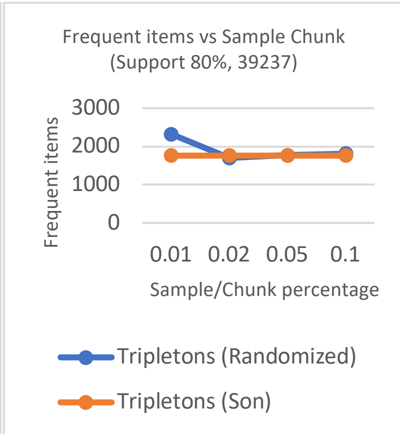
In the following table are described all the experiments performed with Randomized and Son algorithm on pumsb dataset. For different supports {20%,40,60%,80%} were experimented lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: Pumb Dataset size : 49046																
Support	20%	20%	20%	20%	40%	40%	40%	40%	60%	60%	60%	60%	80%	80%	80%	80%
Support count (threshold)	1883	1883	1883	1883	9417	9417	9417	9417	23542	23542	23542	23542	39237	39237	39237	39237
Effective threshold (support count)	19	38	94	188	94	188	471	942	235	471	1177	2354	392	785	1962	3924
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	194	204	193	192	107	108	115	112	55	58	62	59	27	25	25	26
Singletons (Son)	190	190	190	190	112	112	112	112	59	59	59	59	25	25	25	25
Doubletons (Randomized)	10132	10580	9994	10121	3729	3849	4296	4219	1019	1121	1167	1154	317	275	280	283
Doubletons (Son)	9966	9966	9966	9966	4226	4226	4226	4226	1108	1108	1108	1108	276	276	276	276
Tripletons (Randomized)	277475	292602	272424	277893	69005	73659	87552	84971	10717	11830	11960	12210	2324	1699	1774	1814
Tripletons (Son)	274142	274142	274142	274142	84805	84805	84805	84805	11368	11368	11368	11368	1760	1760	1760	1760
Runtime (Randomized) (millisecs)	20836	45121	91363	149043	10256	15415	39203	83467	3904	8646	28405	32120	4172	6837	6792	9741
Runtime (Son) (milliseconds)	2311349	2566706	2532972	2337729	1395781	1433062	1495607	1370119	487101	479654	502919	530272	164583	167866	184429	185477
Memory used (Randomized) (KB)	255236	277597	281830	294632	82074	90097	100403	98511	12637	12635	12635	12635	12635	12635	12635	14564
Memory used (Son) (KB)	291826	291798	291786	291786	98406	98406	98406	98406	12644	12644	12644	12648	16033	16004	15988	15899

In the appendix 4.11 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 80%.







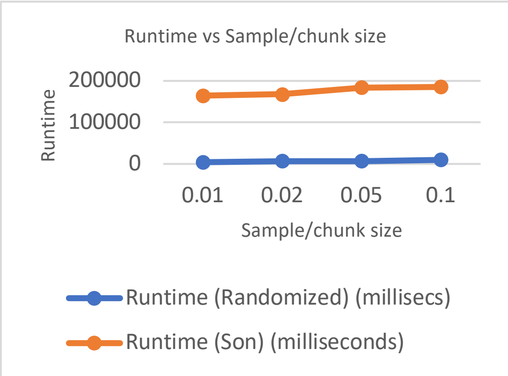
Observations:

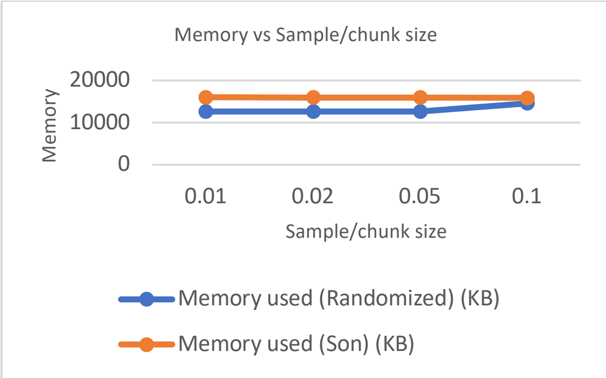
As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise. However, the minimum variation varies between the sample sizes.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 80% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 80% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

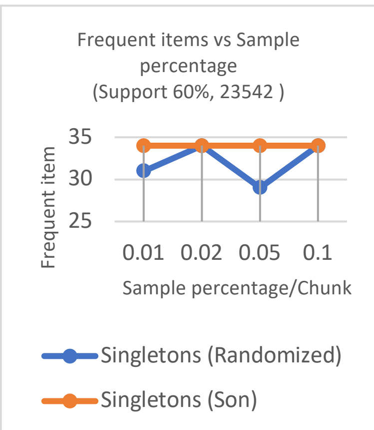
1.4.5 Comparison on pumsb star dataset

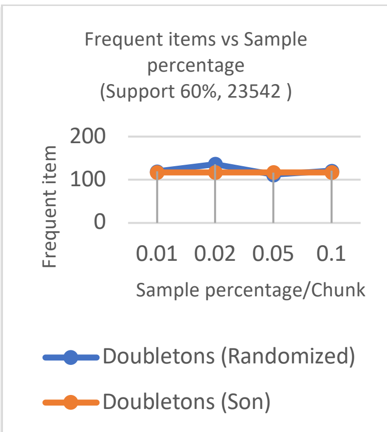
Frequent items

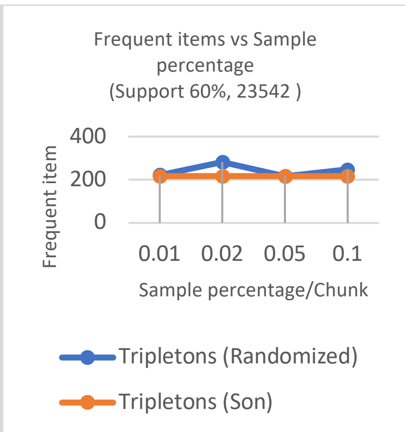
In the following table are described all the experiments performed with Randomized and Son algorithm on pumsb star dataset. For different supports {20%,40,60%,80%} were experimented lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: Pumb_star Dataset size : 49046																
Support	20%	20%	20%	20%	40%	40%	40%	40%	60%	60%	60%	60%	80%	80%	80%	80%
Support count (threshold)	1883	1883	1883	1883	9417	9417	9417	9417	23542	23542	23542	23542	39237	39237	39237	39237
fective threshold (support cour	19	38	94	188	94	188	471	942	235	471	1177	2354	392	785	1962	3924
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	176	168	167	166	87	90	88	89	31	34	29	34	2	0	1	1
Singletons (Son)	165	165	165	165	87	87	87	87	34	34	34	34	0	0	0	0
Doubletons (Randomized)	6052	5744	5715	5642	1828	1893	1748	1827	119	136	112	121	0	0	0	0
Doubletons (Son)	5667	5667	5667	5667	1808	1808	1808	1808	117	117	117	117	0	0	0	0
Tripletons (Randomized)	96784	90979	91408	89797	17034	17890	15253	16667	222	282	216	248	0	0	0	0
Tripletons (Son)	89731	89731	89731	89731	16281	16281	16281	16281	217	217	217	217	0	0	0	0
untime (Randomized) (millisec	6395	8890	18057	34478	2636	4286	8586	16942	1242	1692	2652	4566	623	835	2231	3216
Runtime (Son) (milliseconds)	536069	515109	564323	554383	259617	252773	257989	3E+05	59242	58275	52270	54737	27134	26862	25607	25489
Memory used (Randomized) (KB	114282	114415	121863	124505	12635	37731	12637	12635	13242	13606	13293	13571	11728	11782	11783	11783
Memory used (Son) (KB)	124484	124484	124484	124484	12644	12644	12644	12644	14278	14084	14037	13947	11792	11792	11792	11792

In the appendix 4.12 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 60%.





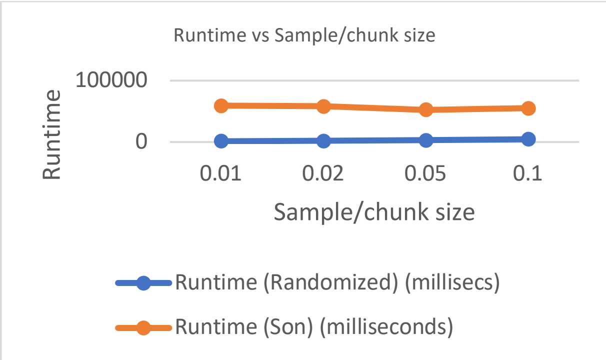


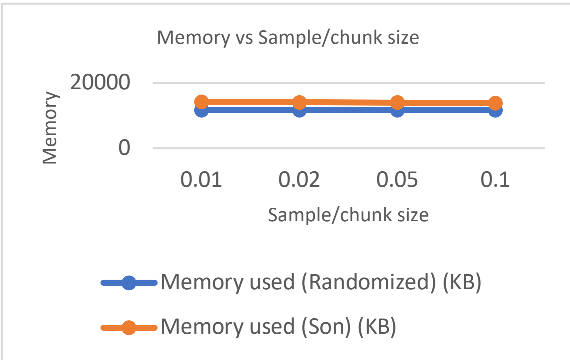
Observations:

- As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.
- If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise. However, the minimum variation varies between the sample sizes.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 60% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 60% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

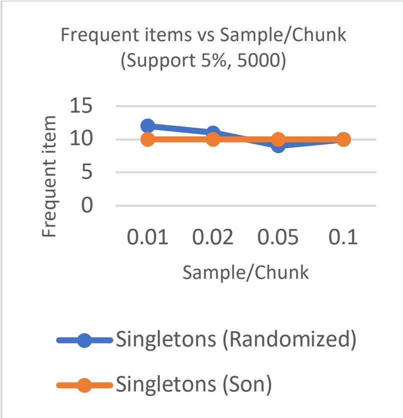
1.4.6 Comparison on T10I4D100K dataset

Frequent items

In the following table are described all the experiments performed with Randomized and Son algorithm on T10I4D100K dataset. For different supports {5%,10%,15%,20%} were experimented with lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: T10I4D100K Dataset size : 100000																
Support	5%	5%	5%	5%	10%	10%	10%	10%	15%	15%	15%	15%	20%	20%	20%	20%
Support count (threshold)	5000	5000	5000	5000	10000	10000	10000	10000	15000	15000	15000	15000	20000	20000	20000	20000
Effective threshold (support count)	50	100	250	500	100	200	500	1000	150	300	750	1500	200	400	1000	2000
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	12	11	9	10	0	0	0	0	0	0	0	0	0	0	0	0
Singletons (Son)	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Runtime (Randomized) (milliseconds)	327	340	746	1064	282	332	601	711	314	322	489	695	335	336	480	766
Runtime (Son) (milliseconds)	3324	3882	3170	3176	2567	2747	2993	3176	2811	2748	2791	2742	2771	2806	3127	3288
Memory used (Randomized) (KB)	11661	11661	11673	11673	11655	11655	11667	11667	11655	11655	11667	11667	11655	11655	11667	11667
Memory used (Son) (KB)	11749	11751	11755	11755	11730	11730	11730	11755	11730	11731	11730	11730	11730	11730	11730	11730

In the following graph is described the experiment in the table, we have only one output, because there are no too many frequent items in the dataset.

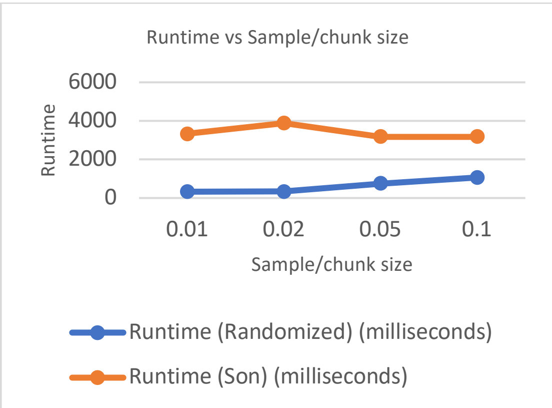


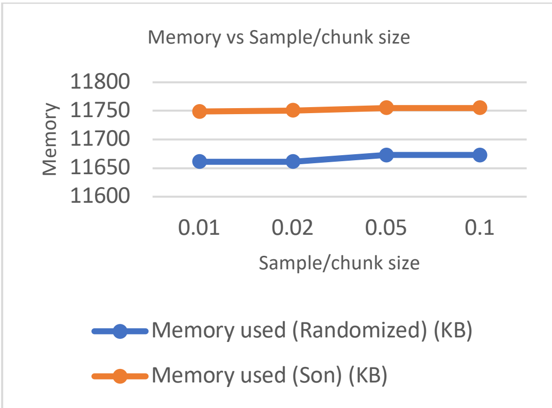
Observations:

- Randomized algorithm has more unstable output than Son algorithm.
- Both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime and memory usage. The following plots show the runtime and memory usage for both algorithms for support threshold 5% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected lower than 5% because we can obtain more number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

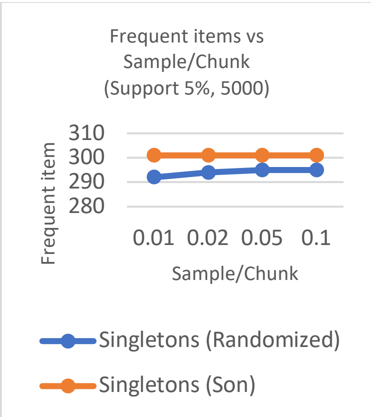
1.4.7 Comparison on T40I10D100K dataset

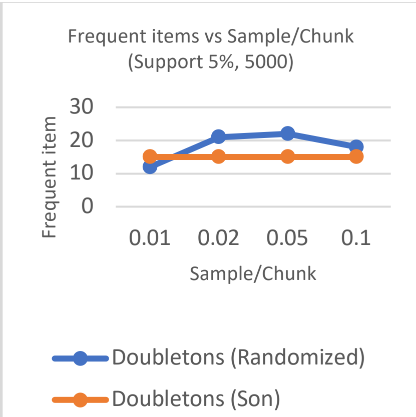
Frequent items

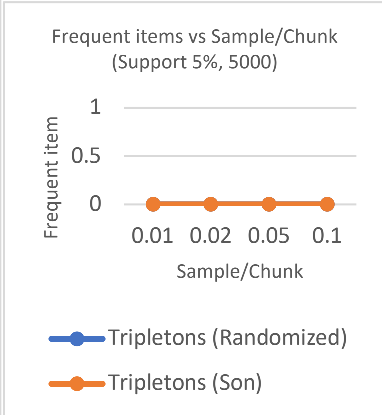
In the following table are described all the experiments performed with Randomized and Son algorithm on T40I10D100K dataset. For different supports {5%,10%,15%,20%} were experimented with lower sample/chunk size {0.1, 0.05, 0.02, 0.01}than exercise 1.4.2 and in the same way than the past exercise singletons size, pairs size, tripletons size, memory usage and run time are obtained.

Dataset: T40I10D100K Dataset size : 100000																
Support	5%	5%	5%	5%	10%	10%	10%	10%	15%	15%	15%	15%	20%	20%	20%	20%
Support count (threshold)	5000	5000	5000	5000	10000	10000	10000	10000	15000	15000	15000	15000	20000	20000	20000	20000
Effective threshold (support count)	50	100	250	500	100	200	500	1000	150	300	750	1500	200	400	1000	2000
Sample/Chunk	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1	0.01	0.02	0.05	0.1
Singletons (Randomized)	292	294	295	295	78	89	80	80	20	17	20	19	3	4	5	4
Singletons (Son)	301	301	301	301	82	82	82	82	19	19	19	19	5	5	5	5
Doubletons (Randomized)	12	21	22	18	0	0	0	0	0	0	0	0	0	0	0	0
Doubletons (Son)	15	15	15	15	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Randomized)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tripletons (Son)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Runtime (Randomized) (milliseconds	3730	10117	21541	44277	1282	2664	4069	5784	1033	1846	2284	4336	1162	1570	2035	4732
Runtime (Son) (milliseconds)	254829	309759	18462	359986	54283	52137	52314	51906	35190	36210	34928	35289	34512	29156	31282	34205
Memory used (Randomized) (KB)	152667	278515	278533	278530	15945	11661	22507	22507	11818	11806	11816	11815	11729	11730	11731	11730
Memory used (Son) (KB)	152823	278585	278539	278557	22521	22519	22519	22521	11826	11826	11828	11826	11744	11744	11742	11742

In the appendix 4.12 there are graphs which describe the experiments in the table for each support by plotting singletons, doubletons and tripletons for each algorithm and sample size. Consider rows are the support and columns are singletons, pairs and tripletons. The following plot shows singletons, doubletons and tripletons for each algorithm and for each sample size for a support of 5%.







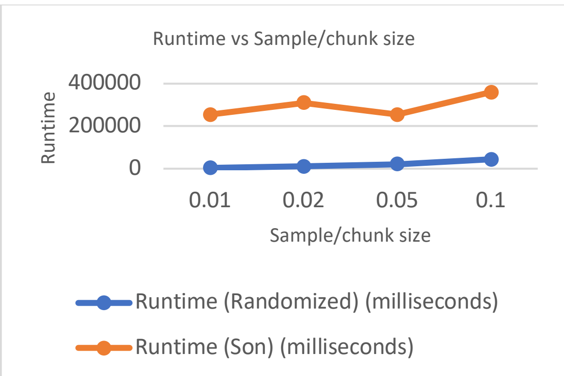
Observations:

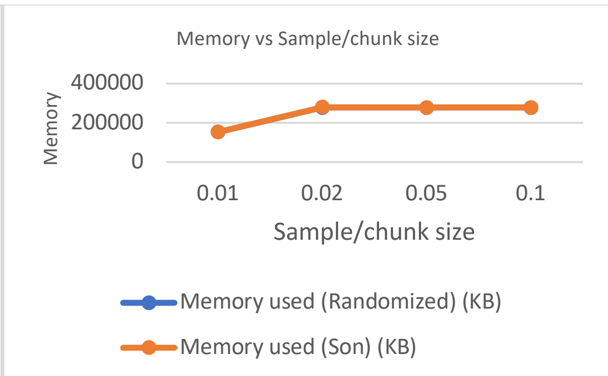
- As we can see in the table and graphs, Randomized algorithm has more unstable output than Son algorithm.

- If the sample is large enough, other words approaching to 1, in the most of the cases, both algorithm has less variation in term of frequent item set such as singleton, pairs and tripletons with respect to larger samples sizes in the previous exercise. However, the minimum variation varies between the sample sizes.

Runtime and memory usage

Randomized algorithm outperform Son Algorithm in terms of runtime however the memory usage is quite similar. The following plots show the runtime and memory usage for both algorithms for support threshold 5% and its different sample/chunk sizes.





Decisions

According to the data provided, assuming that we don't know additional information about association rules or the problem that need to be solved or answered, the support should be selected is at least 5% because we can obtain a manageable number of frequent item. Even though, the randomized algorithm is much more efficient in term of run time and memory, We select the Son algorithm because give more accurate outputs than the randomized algorithm.

1.4.8 General Discussion

- Randomized algorithm has huge variation in term of singletons, doubletons, tripletons, which suggest that the algorithm identify false negatives and false positives.

- The implementation of Randomized algorithm is more efficient in term of memory usage and runtime, since this implementation read the sample from disk only one time for both passes and use a sample of the dataset.

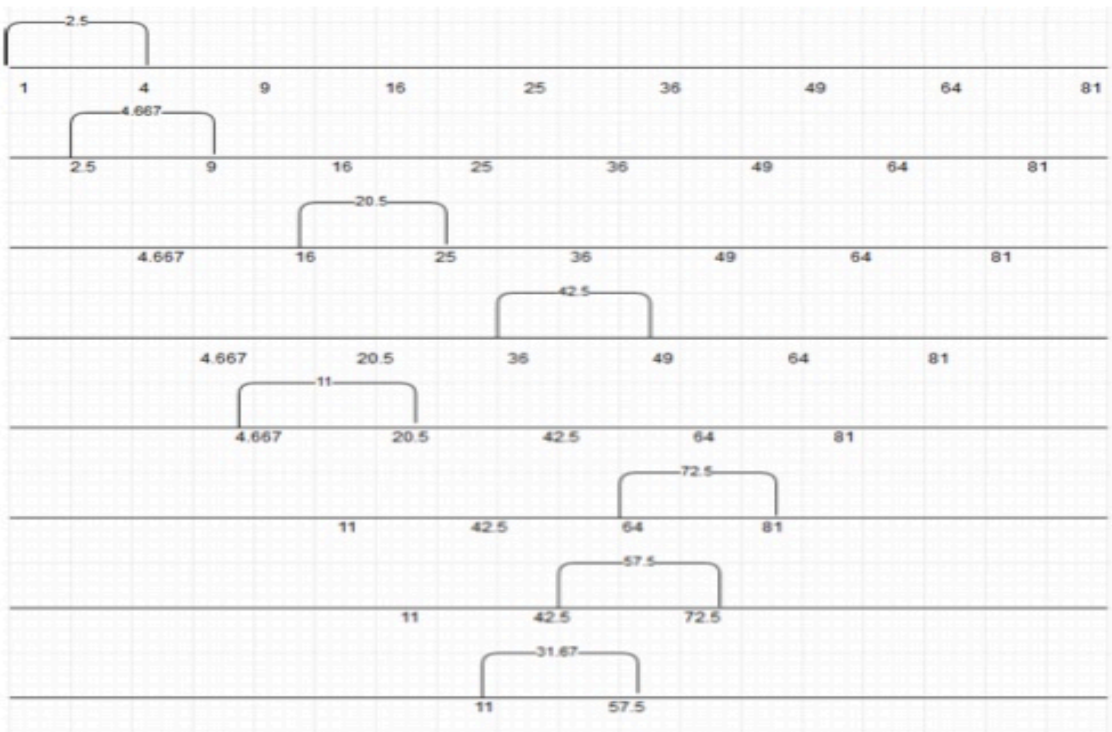
- Samples sizes lower than 0.1 does not mean more false positives and false negatives, there are many cases showed previously where randomized algorithm produce the same output than the Son algorithm, which suggest than we can use randomized algorithm for a particular problem by using less memory and in less time.

1.4.9 Challenges observed in the implementation

For most of the cases, and specially dataset with huge number of transactions such as 100.000, the main challenge is when the effective support threshold is decreasing, the runtime and memory usage increase and sometimes the script crash it, due to segment fault, however, one single experiment can be run in the terminal alone. Another challenge is the exercise is quite extensive, because there a lot of dataset involved as well variables that we need to measure such as singletons, doubletons, tripletons, effective threshold, run time, memory and so on.

2. Exercise 2 Clustering

2.1 Hierarchical clustering

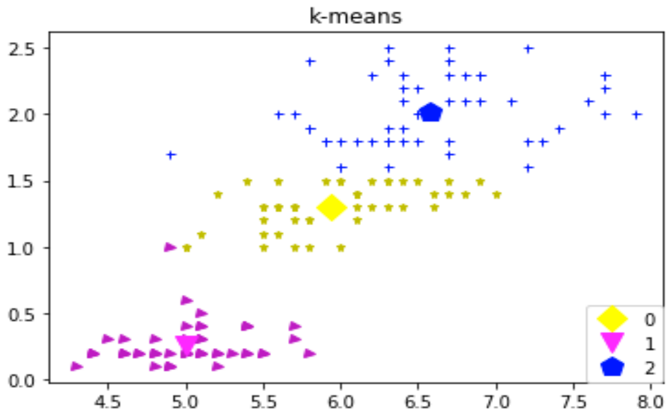


Assuming the clusters are represented by their centroid (average), and at each step the clusters with the closest centroids are merged.

2.2 K-means algorithm

2.2.1 Plot k-means results

Plot two dimensions for k=3 and its centroids

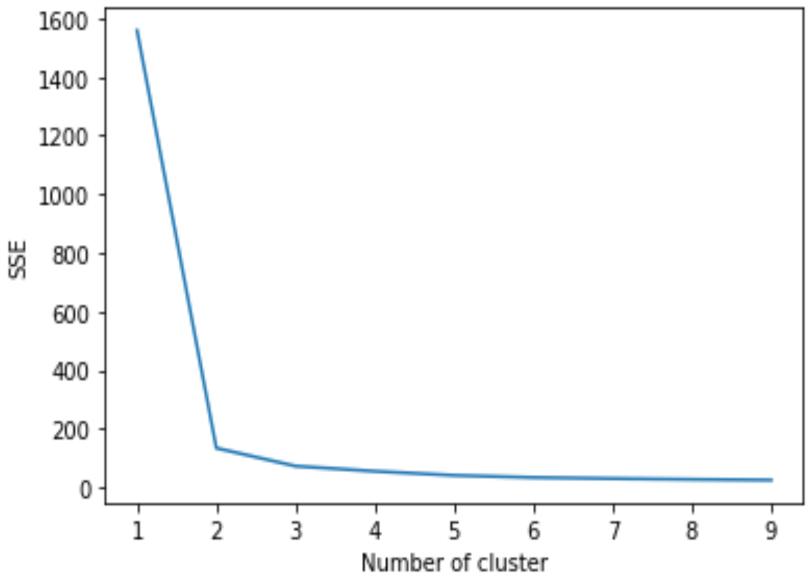


2.2.2 Picking k value

There are numbers of methods in the literature for identifying the optimal number of clusters. I picked only one method to select k. The method computes the sum of squared error (SSE) for some values of k. The SSE is defined as the sum of the squared distance between each member of the cluster and its centroid. Mathematically:

$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(x, c_i)^2$$

Plotting k against the SSE (code attached), we can observe that the error decreases as k gets larger; this is because when the number of clusters increases, they should be smaller. The idea is to choose the k at which the SSE decreases abruptly which is k=3 for this case.



Since Iris dataset contains three different types of flowers, it is not surprising to end up with 3 clusters being the most effective method for this case.

3. Exercise 3 Advertising

3.1 Question 1

Settings:
query sequence xxyyzz
Each advertiser has a budget of 2.
Advertiser A only bids on x, B bids on x and y, and C bids on x, y, and z.

Solution:
Since query x can be assigned to X or Y or Z, both x queries will be assigned. The following two scenarios will happen:

1. The budget of one specific advertiser will be zero

2. The budget of two advertiser reduce by 1 unit

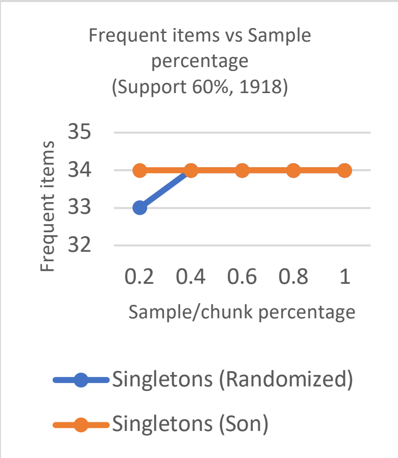
Since y can be assigned to 2 advertisers, in both of the cases discussed above, there will be at least total budget of 2 remaining for the advertisers that can take in y. So, both y will be assigned eventually. So, at least 4 of these 6 queries will be assigned in any case.

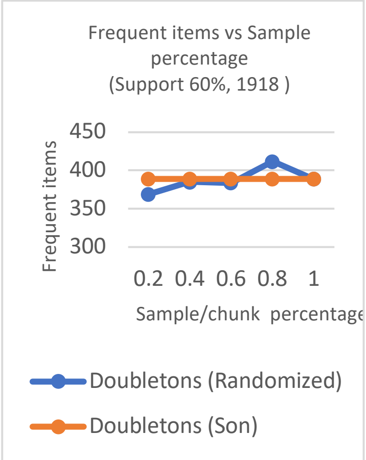
3.2 Question 2

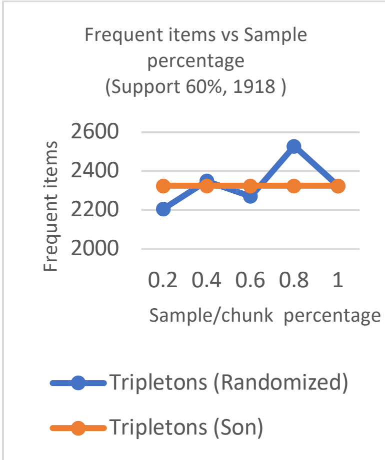
The query can be xxzz for a specific case, xx will be assigned to C and thus zz will remain unassigned. Since optimal offline algorithm can assign all 4 of these queries and our greedy might assign only 2 out of these queries, the ratio will be 2/4 = ½ .Other queries like yyzz can exist too.

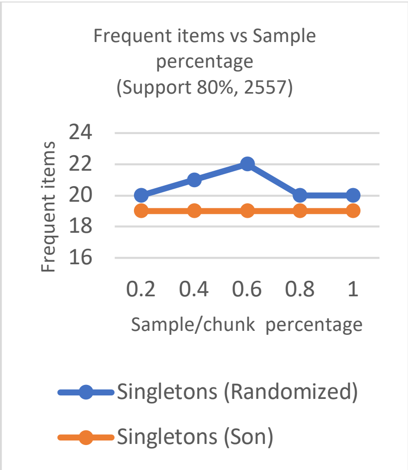
4. Appendix

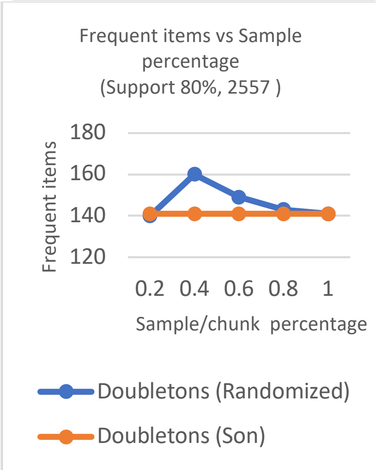
4.1 Chess plots sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

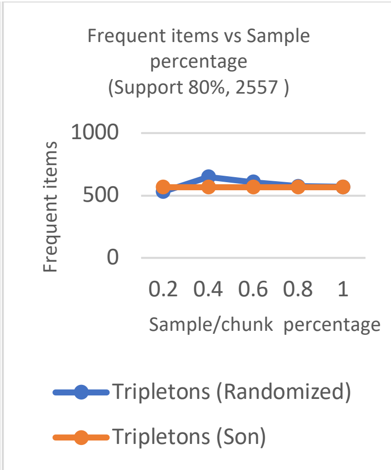




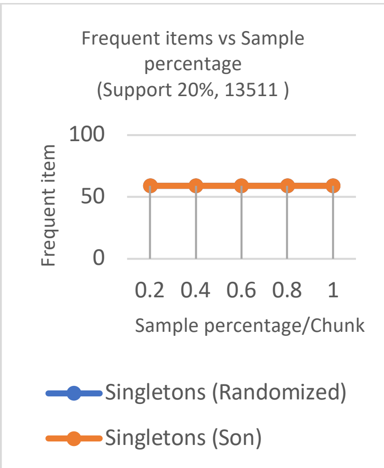


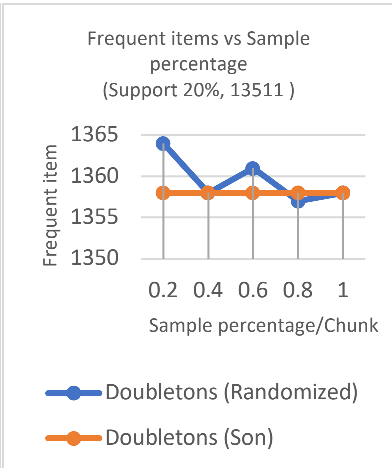


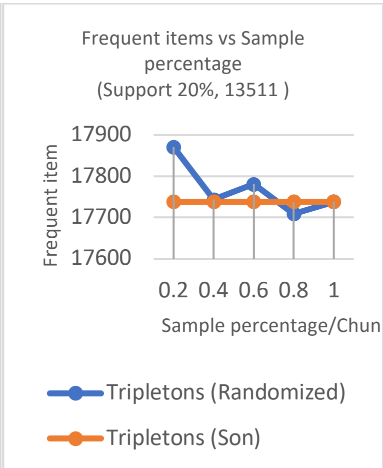


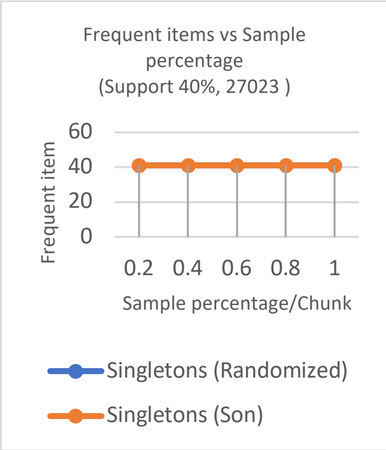


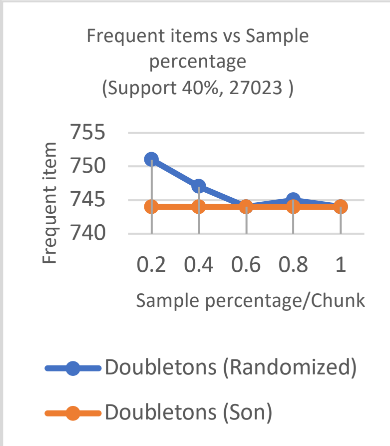
4.2 Connect plots sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

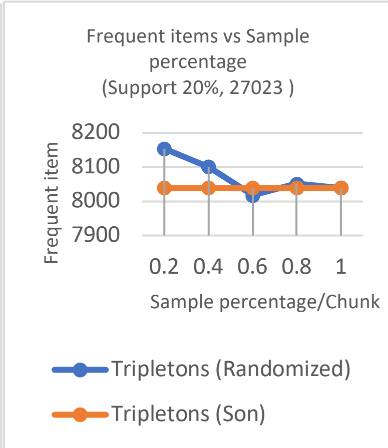


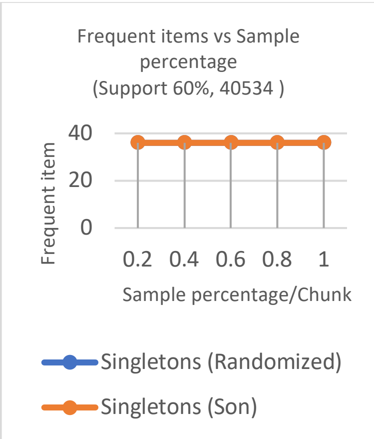


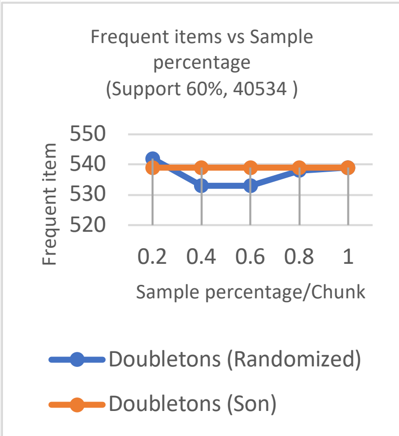


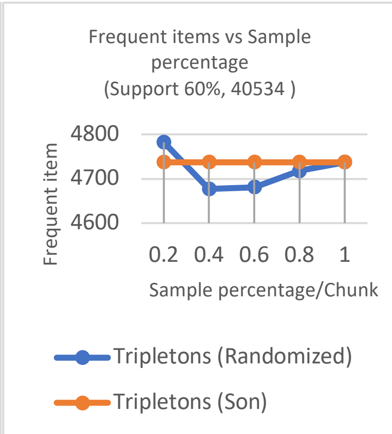


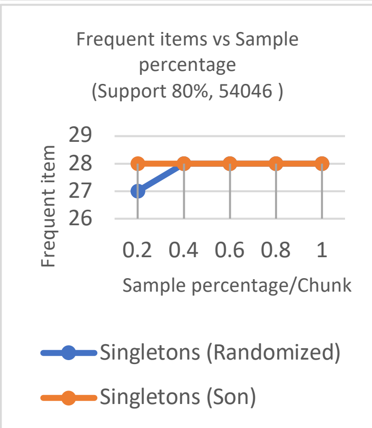


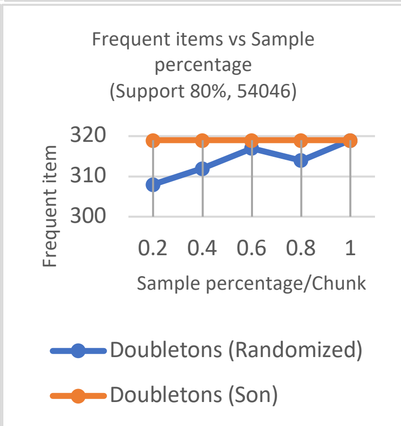


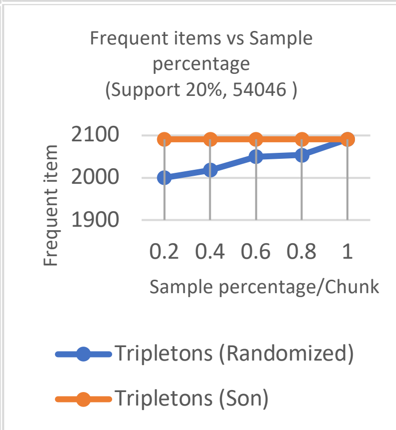










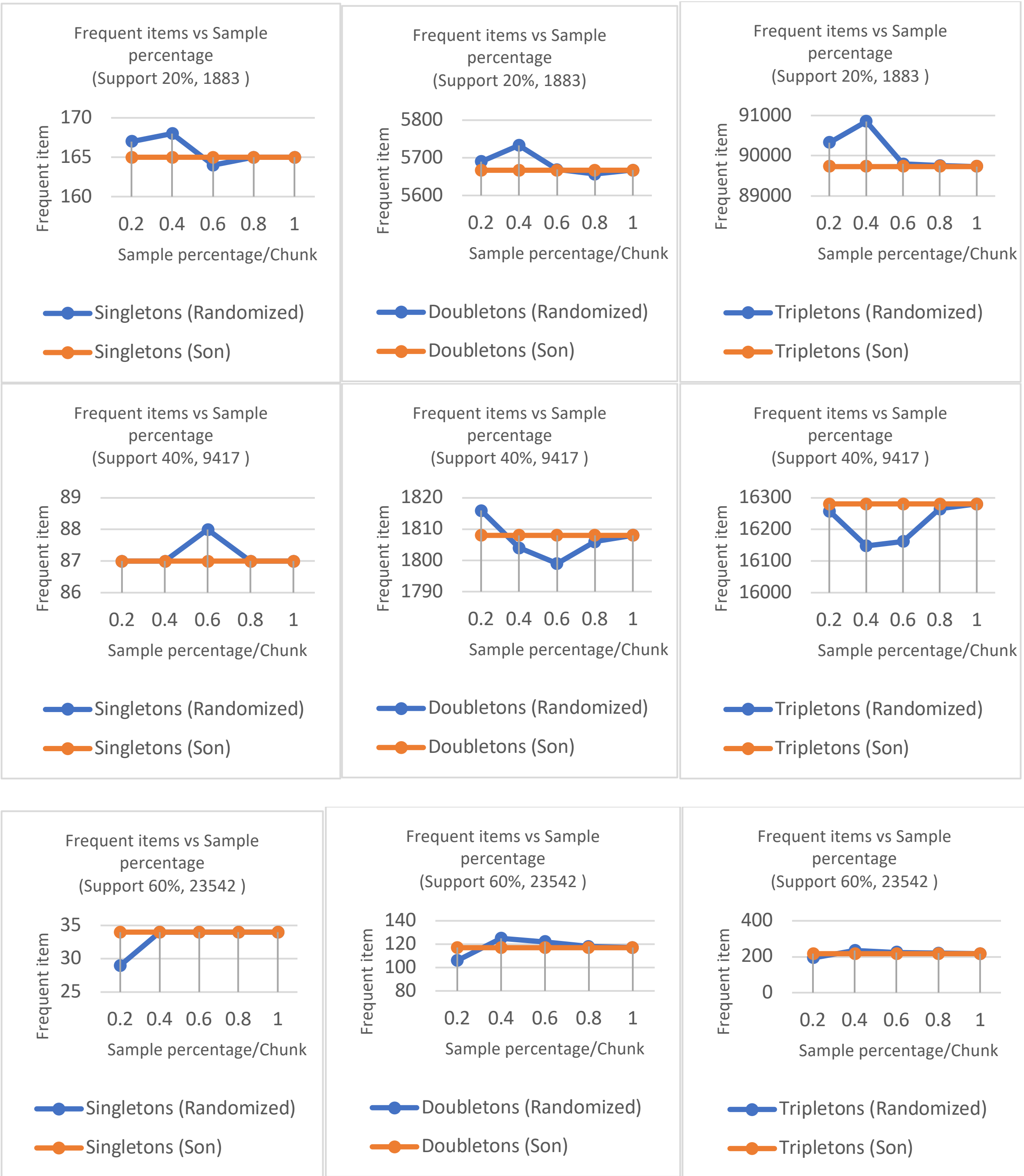


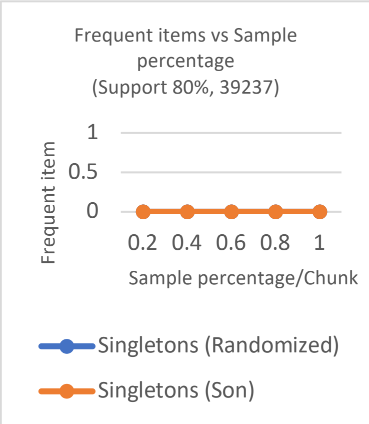
4.3 Mushroom plots sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

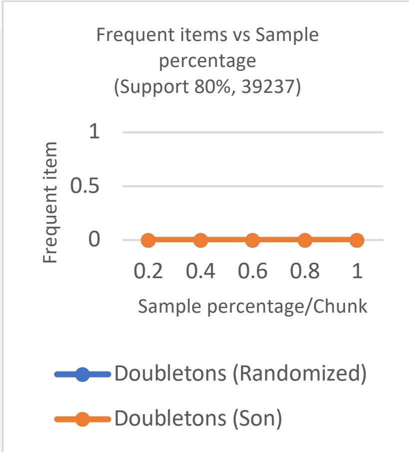
4.4 **Pumsb plots sample/chunk size {0.2, 0.4,0.6, 0.8, 1}**

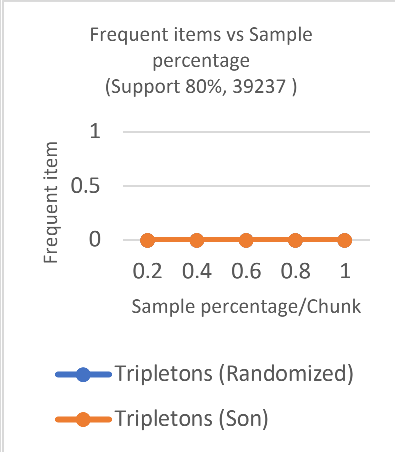

Frequent items vs Sample Chunk (Support 20%, 1883)
Frequent items
196
194
192
190
188
0.2 0.4 0.6 0.8 1
Sample/Chunk percentage
Singletons (Randomized)
Singletons (Son)
Frequent items vs Sample Chunk (Support 20%, 1883)
10100
10050
10000
9950
9900
Doubletons (Randomized)
Doubletons (Son)
Frequent items vs Sample Chunk (Support 20%, 1883)
277000
276000
275000
274000
273000
Tripletons (Randomized)
Tripletons (Son)
Frequent items vs Sample Chunk (Support 40%, 9417)
150
100
50
0
Singletons (Randomized)
Singletons (Son)
Frequent items vs Sample Chunk (Support 40%, 9417)
4240
4230
4220
4210
Doubletons (Randomized)
Doubletons (Son)
Frequent items vs Sample Chunk (Support 40%, 9417)
86000
85500
85000
84500
84000
Tripletons (Randomized)
Tripletons (Son)
Frequent items vs Sample Chunk (Support 60%, 23542)
60
59
58
57
56
Singletons (Randomized)
Singletons (Son)
Frequent items vs Sample Chunk (Support 60%, 23452)
1150
1100
1050
1000
Doubletons (Randomized)
Doubletons (Son)
Frequent items vs Sample Chunk (Support 60%, 23542)
12000
11500
11000
10500
10000
Tripletons (Randomized)
Tripletons (Son)
Frequent items vs Sample Chunk (Support 80%, 39237)
30
20
10
0
Singletons (Randomized)
Singletons (Son)
Frequent items vs Sample Chunk (Support 80%, 39237)
280
278
276
274
272
Doubletons (Randomized)
Doubletons (Son)
Frequent items vs Sample Chunk (Support 80%, 39237)
1800
1750
1700
1650
Tripletons (Randomized)
Tripletons (Son)

4.5 Pumsb star plots sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

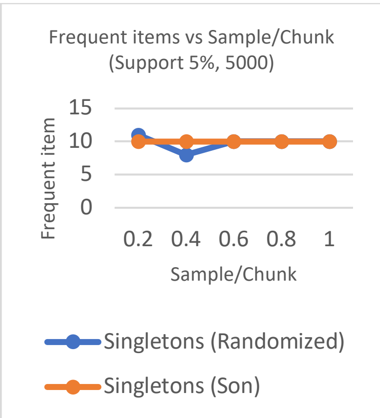


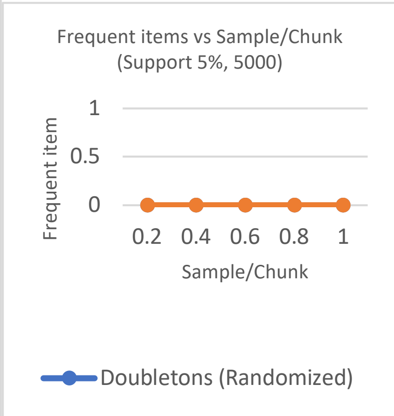


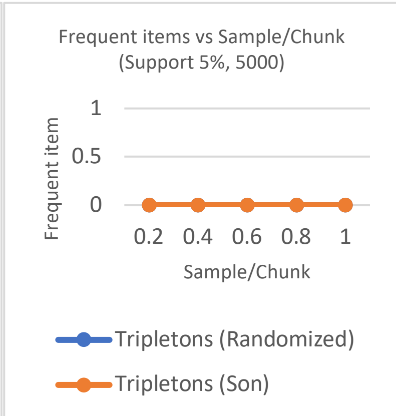


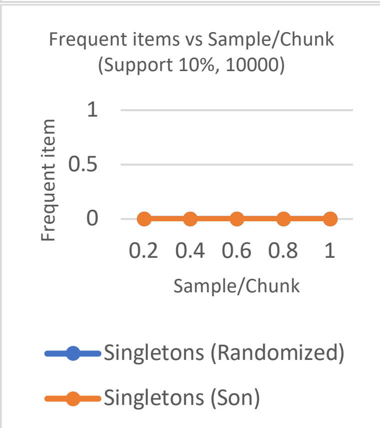


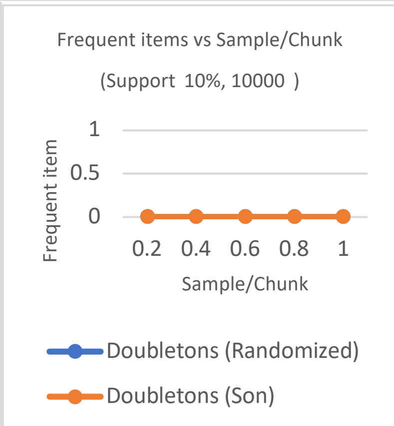
4.6 Plots T10I4D100K sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

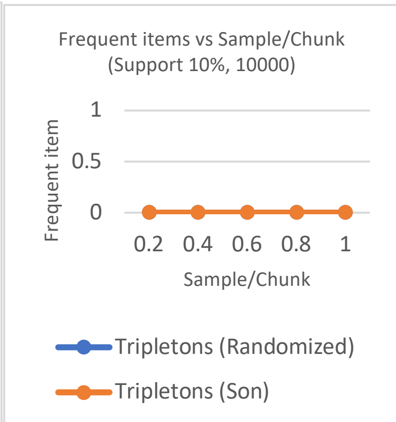


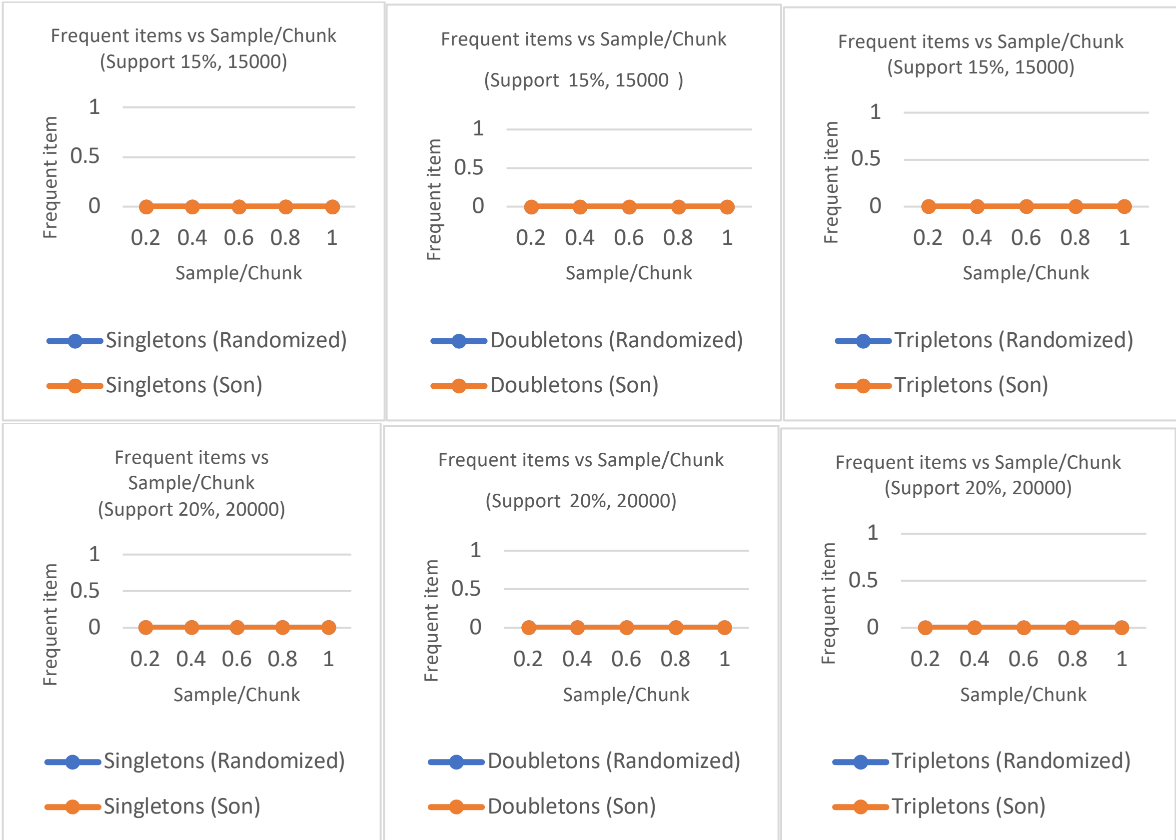




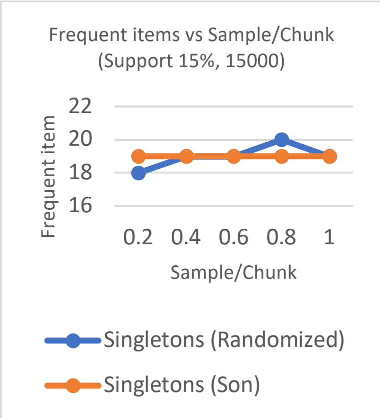


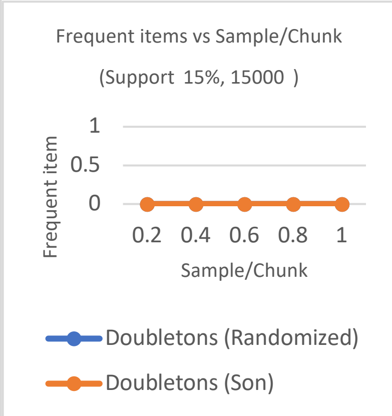


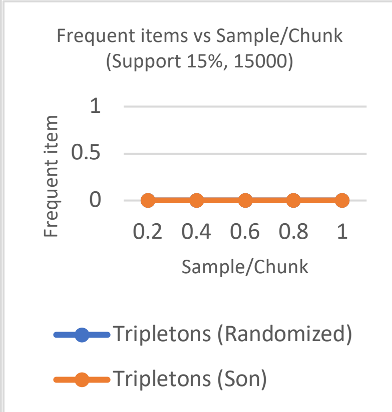


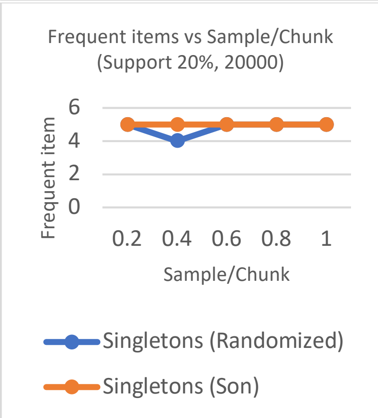


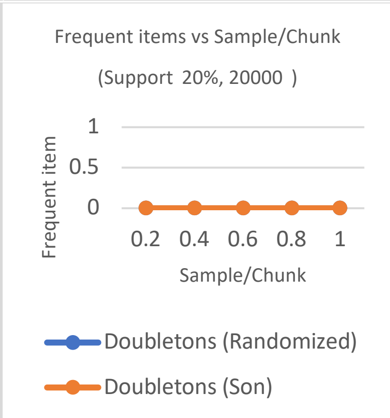
4.7 Plots T40I10D100K sample/chunk size {0.2, 0.4,0.6, 0.8, 1}

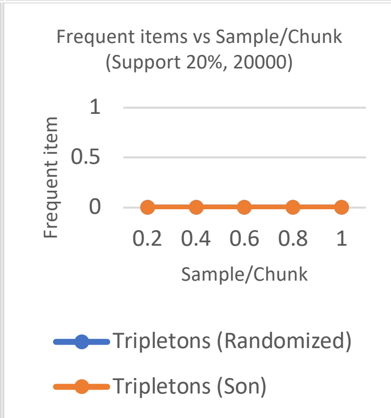




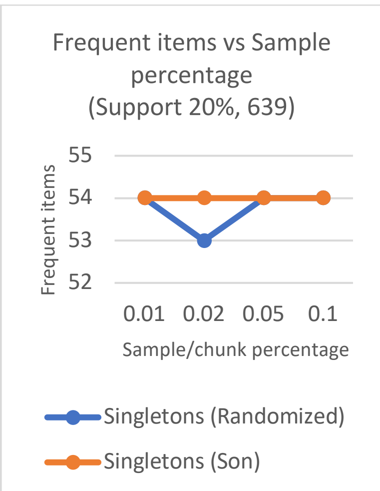


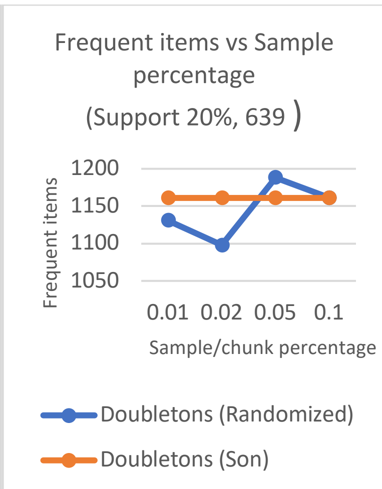


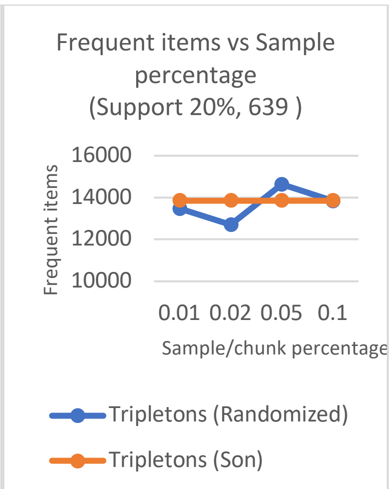


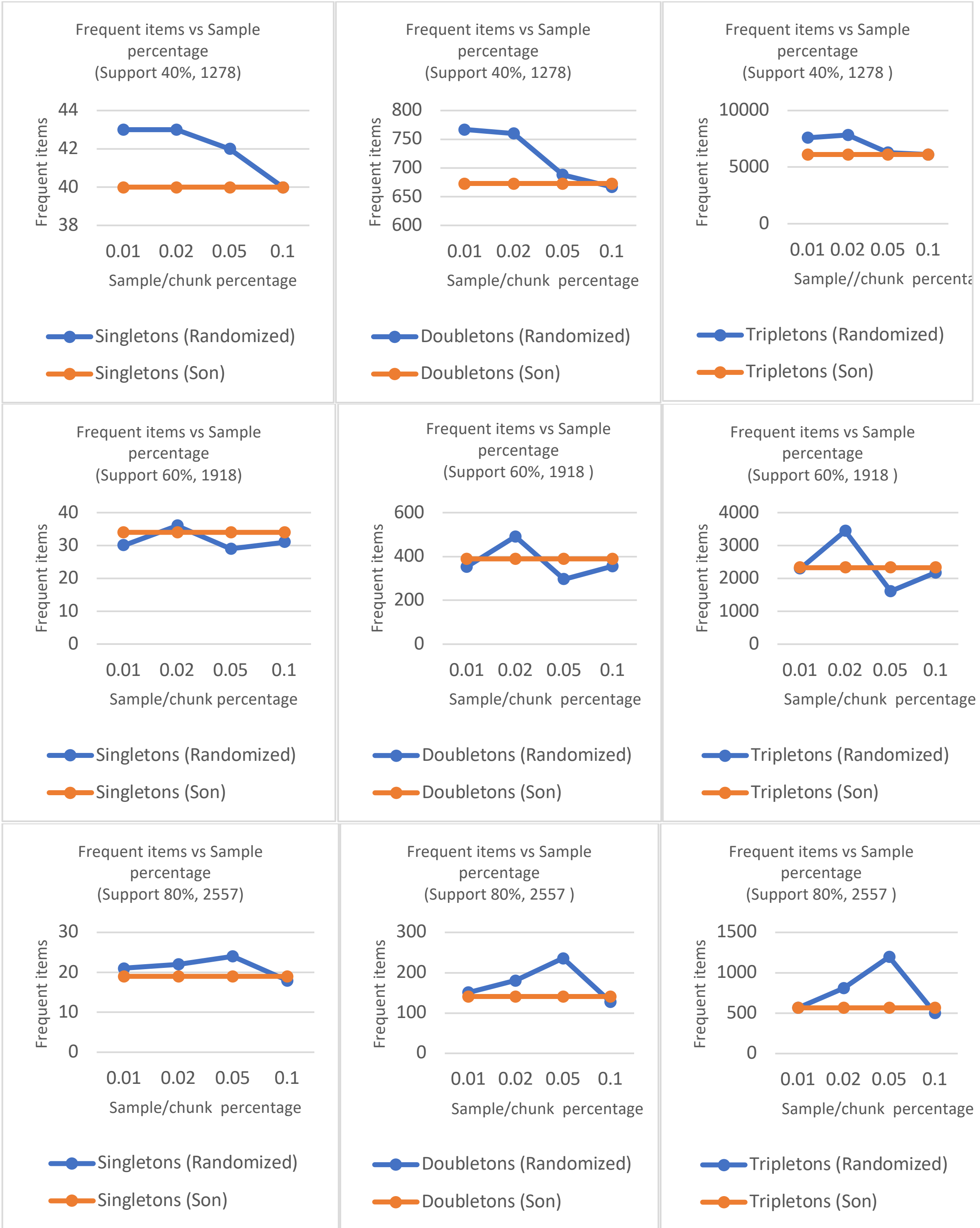


4.8 Chess plots sample/chunk size {0.01, 0.02,0.05, 0.1}

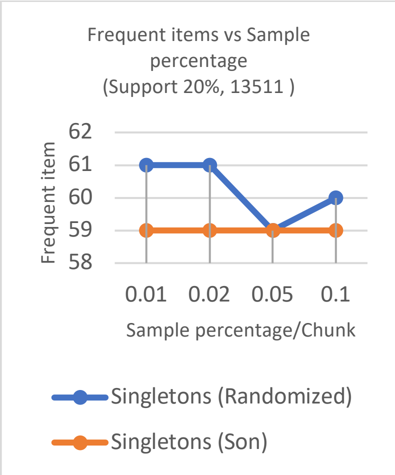


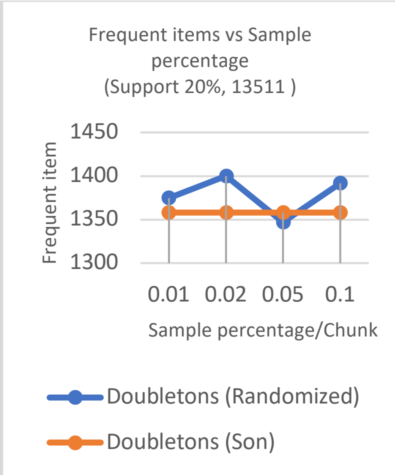


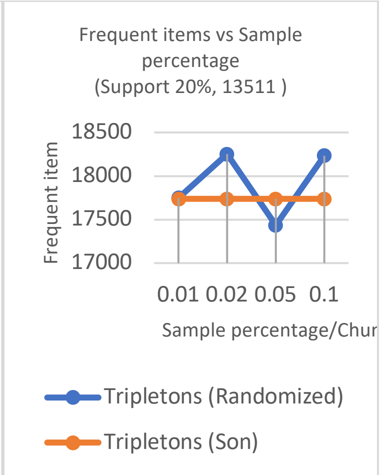


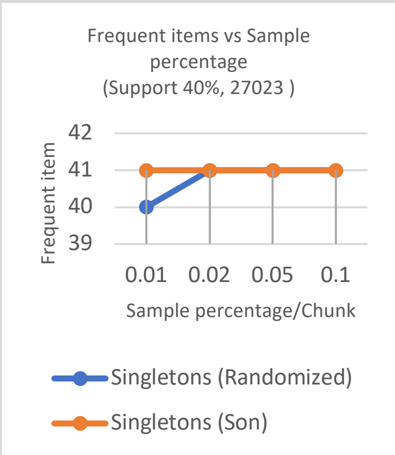


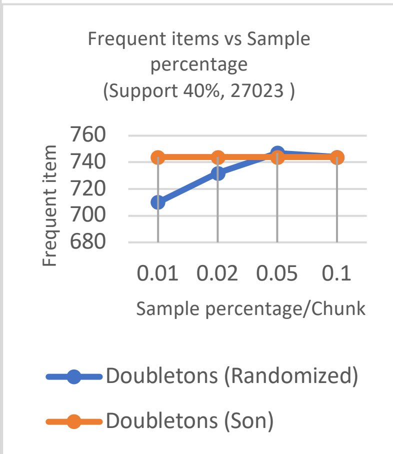
4.9 Connect plots sample/chunk size {0.01, 0.02,0.05, 0.1}

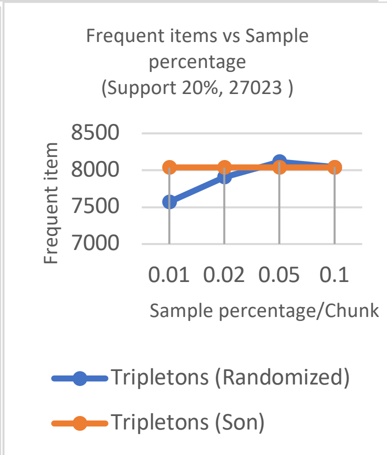

Frequent items vs Sample percentage (Support 20%, 13511)
Frequent item
62
61
60
59
58
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Singletons (Randomized)
Singletons (Son)

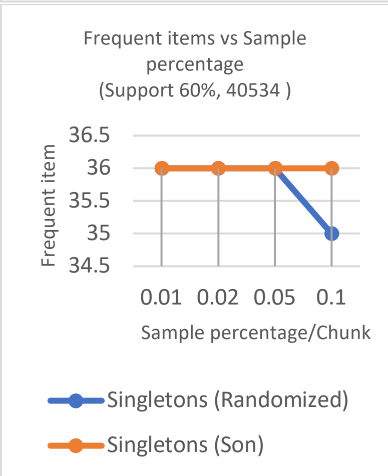

Frequent items vs Sample percentage (Support 20%, 13511)
Frequent item
1450
1400
1350
1300
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Doubletons (Randomized)
Doubletons (Son)

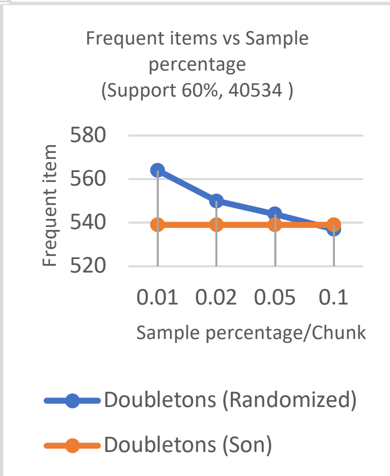

Frequent items vs Sample percentage (Support 20%, 13511)
Frequent item
18500
18000
17500
17000
0.01 0.02 0.05 0.1
Sample percentage/Chur
Tripletons (Randomized)
Tripletons (Son)

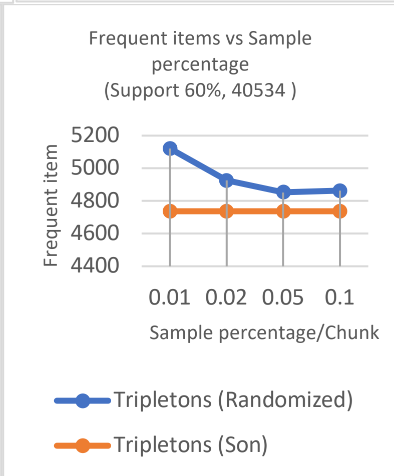

Frequent items vs Sample percentage (Support 40%, 27023)
Frequent item
42
41
40
39
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Singletons (Randomized)
Singletons (Son)

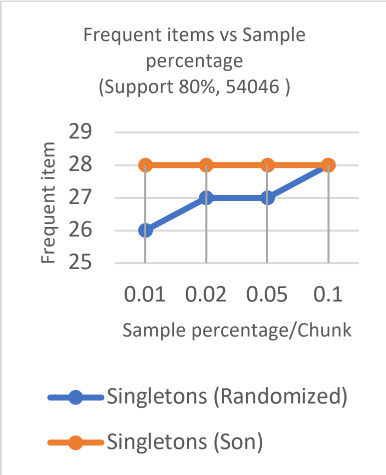

Frequent items vs Sample percentage (Support 40%, 27023)
Frequent item
760
740
720
700
680
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Doubletons (Randomized)
Doubletons (Son)

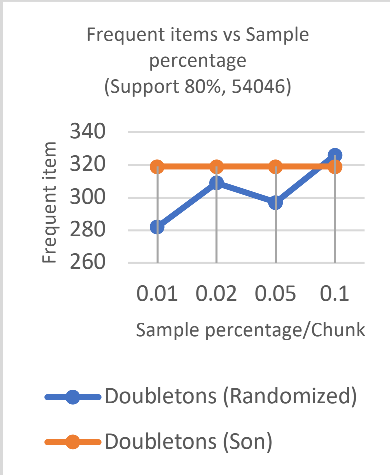

Frequent items vs Sample percentage (Support 20%, 27023)
Frequent item
8500
8000
7500
7000
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Tripletons (Randomized)
Tripletons (Son)

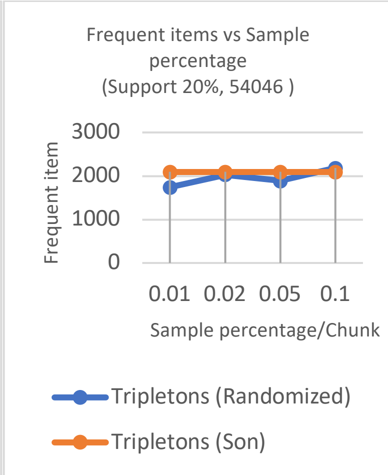

Frequent items vs Sample percentage (Support 60%, 40534)
Frequent item
36.5
36
35.5
35
34.5
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Singletons (Randomized)
Singletons (Son)


Frequent items vs Sample percentage (Support 60%, 40534)
Frequent item
580
560
540
520
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Doubletons (Randomized)
Doubletons (Son)

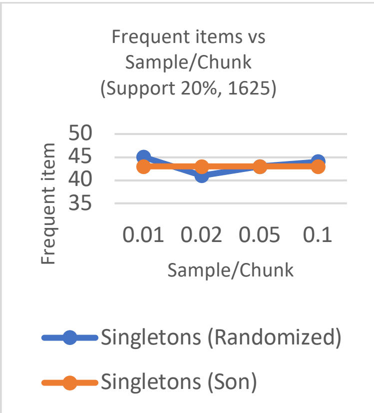

Frequent items vs Sample percentage (Support 60%, 40534)
Frequent item
5200
5000
4800
4600
4400
0.01 0.02 0.05 0.1
Sample percentage/Chunk
Tripletons (Randomized)
Tripletons (Son)

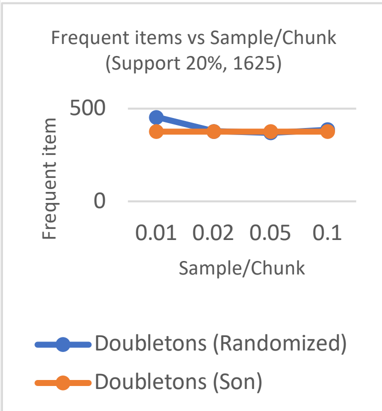


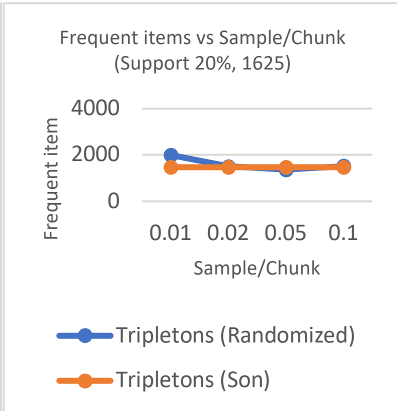


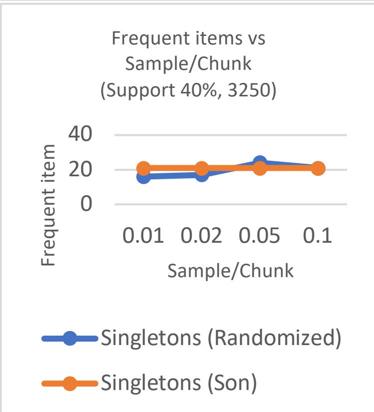


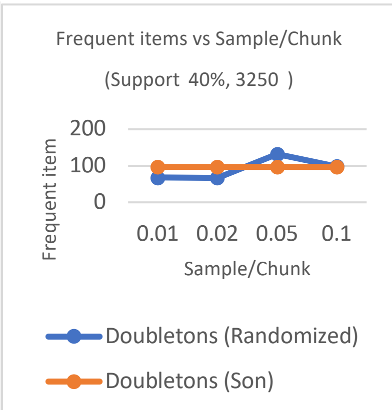
4.10 Mushroom plots sample/chunk size {0.01, 0.02,0.05, 0.1}

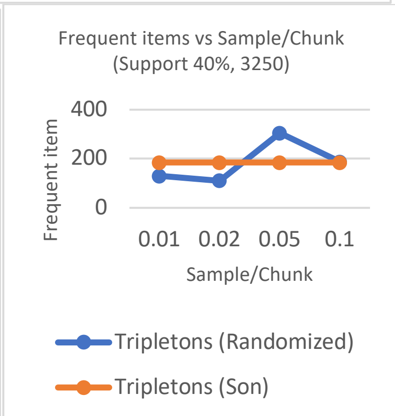


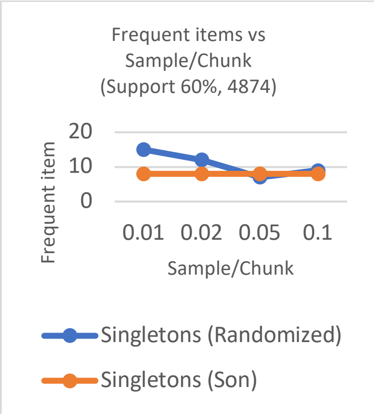


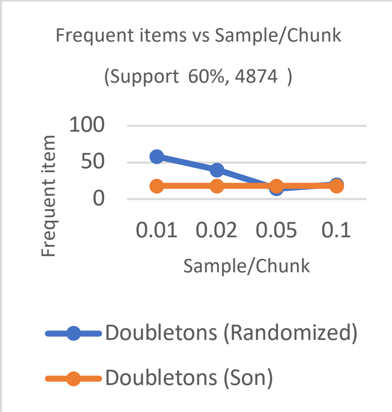


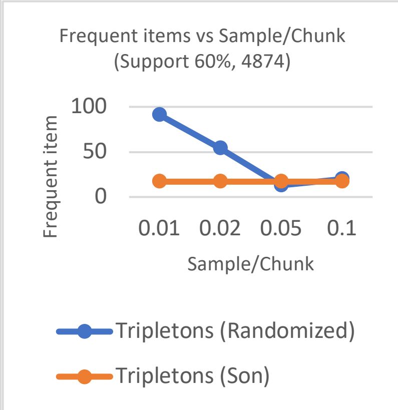


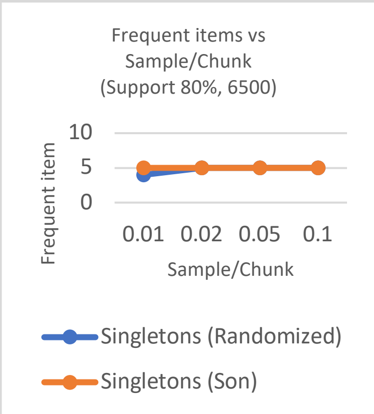


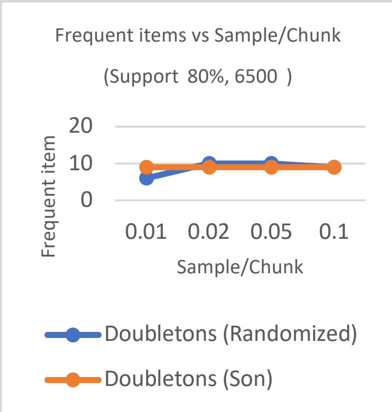


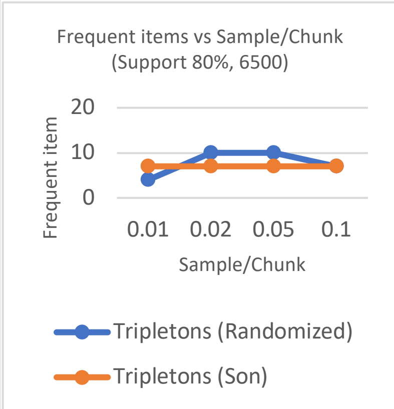




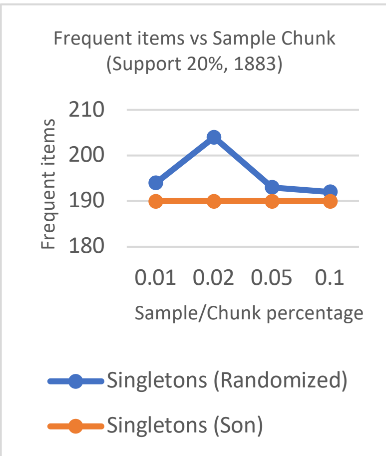


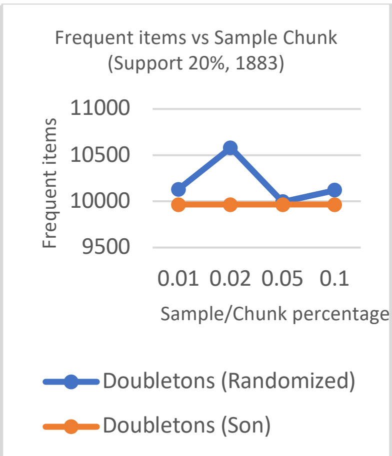


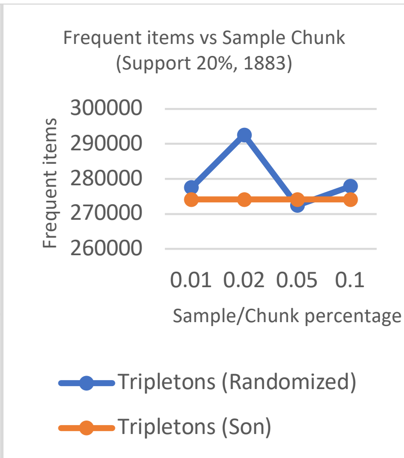


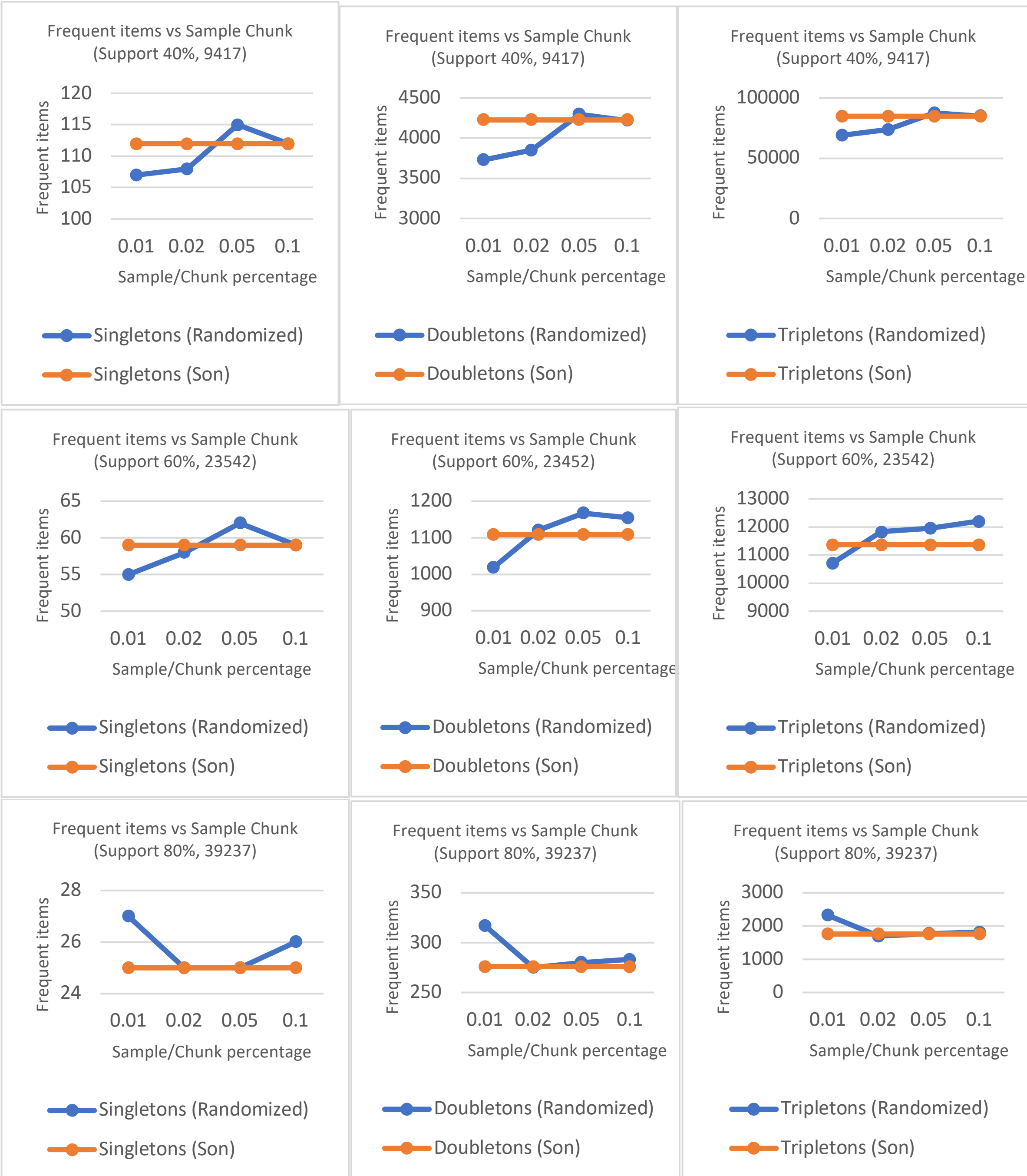


4.11 Pumsb plots sample/chunk size {0.01, 0.02,0.05, 0.1}

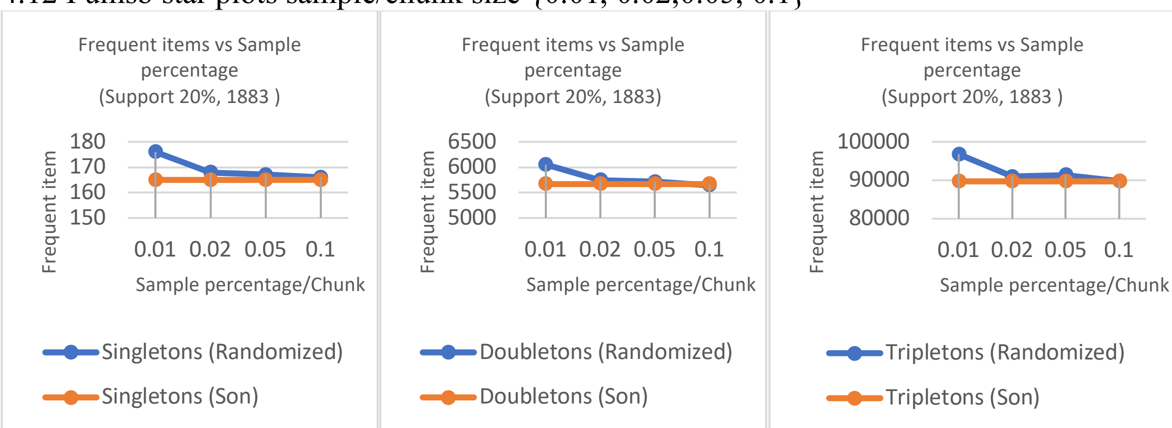




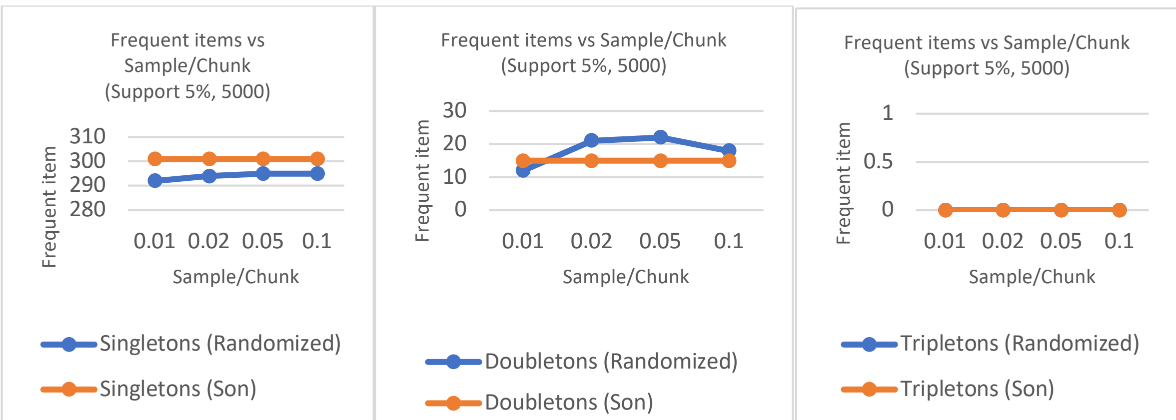


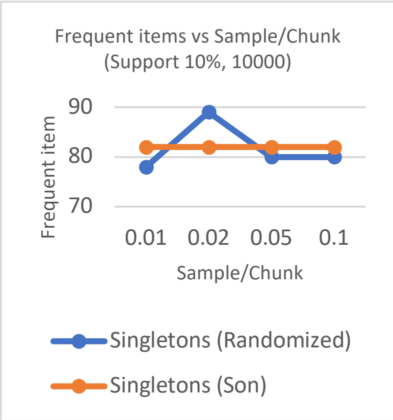


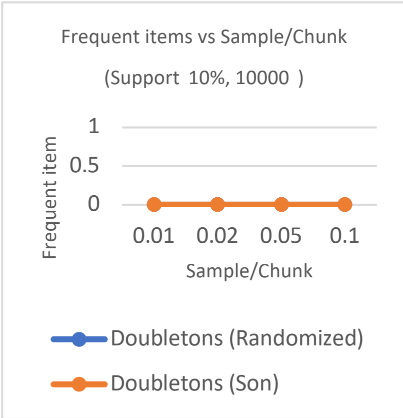
4.12 Pumsb star plots sample/chunk size {0.01, 0.02,0.05, 0.1}

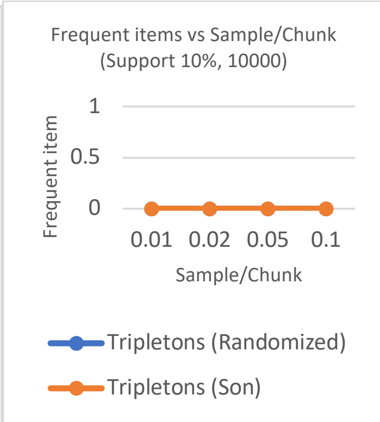


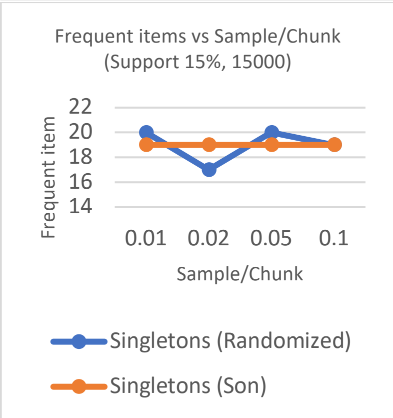
4.14 Plots T40I10D100K sample/chunk size {0.01, 0.02,0.05, 0.1}

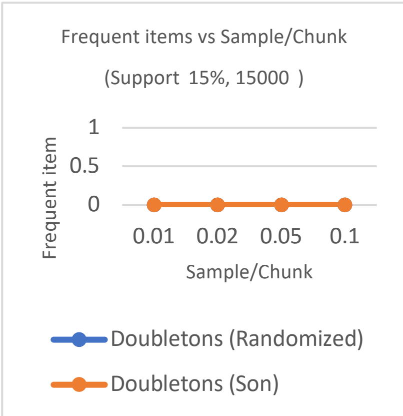


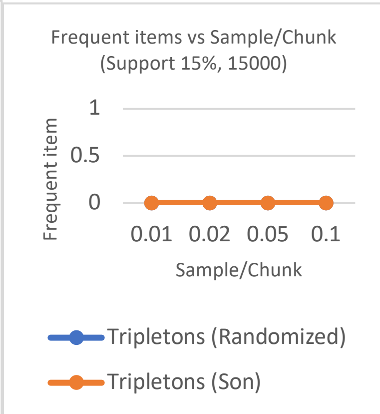


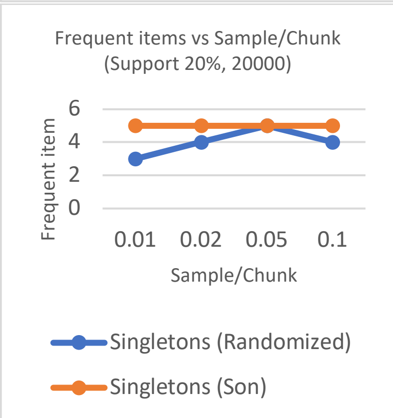


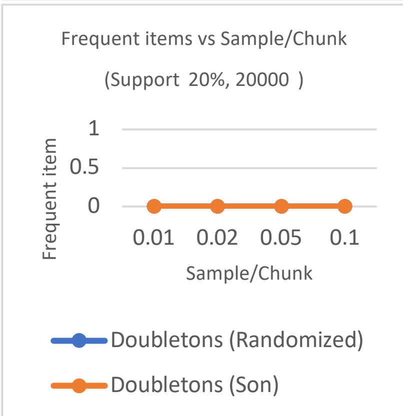


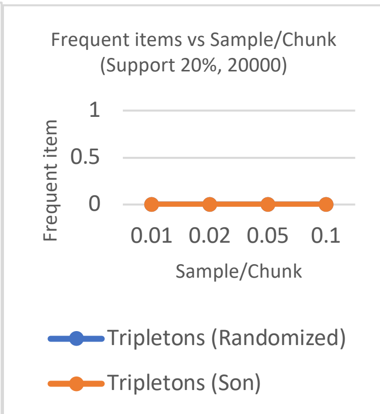












5. References

Mining Big Data Lectures

Anand Rajaraman and Jeffrey Ullman, Mining of Massive Datasets