A

Dummy Generation for MedCo

MARCO ZOVERALLI, EPFL

1. INTRODUCTION

The dummy generation is achieved by duplicating real patients and by shuffling their ontology codes, in such way that at the end the number of occurrences of each code (i.e. the number of patients that have a certain ontology code) is the same. As shown in [1], this procedure can be excessively expensive, and therefore an alternative approach is considered: we divide ontology codes into clusters, and we ensure that the frequency of each code within the cluster is the same. This guarantees that less dummy patients need to be added. On the other hand, this also reduces the anonymity of the patients. In particular, the tradeoff of this technique consists in reducing the maximum ratio (i.e. how much the database with dummies is larger in comparison with the original one) while keeping the minimum anonymity set (i.e. the size of the smallest cluster) at an acceptable level.
Therefore, the dummy generation process consists in two phases:

— codes clustering
— generation of dummy patients

The implementation can be found on https://c4science.ch/source/import-tool/

2. CLUSTERING

During the first phase, one important requirement is that highly correlated codes end up in the same bucket. In this context, two codes are correlated if they appear in the same patient several times. This characteristic is properly caught by the Jaccard index, which is the similarity metric that was used during this clustering phase. When clustering occurs, a distance metric needs to be considered (which can be easily computed from the any similarity coefficient).
Correlation (or any other form of relative similarity) is not a euclidean distance. This means that the distance cannot be based on the locations of the points in a space, but it is based on some properties of points. This implies that traditional clustering techniques (such as k-means) need to be used carefully and adapted accordingly (for instance, there is no concept of average between any two points). Among the techniques that perform clustering based on (dis)similarity, hierarchical agglomerative clustering (HAC [2]) is quite relevant and it was used for this phase.
Turning to the application of such techniques on the demo dataset, further considerations needed to be taken into account. Below, an histogram representing the frequency of ontology codes is shown:

[1]https://infoscience.epfl.ch/record/232605
[2]https://en.wikipedia.org/wiki/Hierarchical_clustering

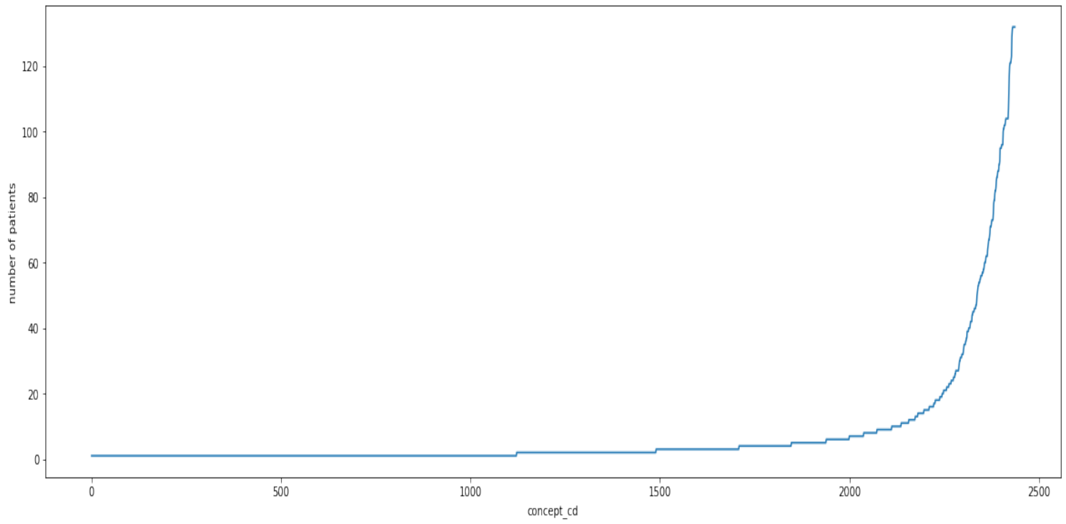

Fig. 1. Histogram

where the x axis represent the different ontology codes and the y axis represents the number of patients associated to each code.
In the ideal case, all the codes should have the same number of occurrences:

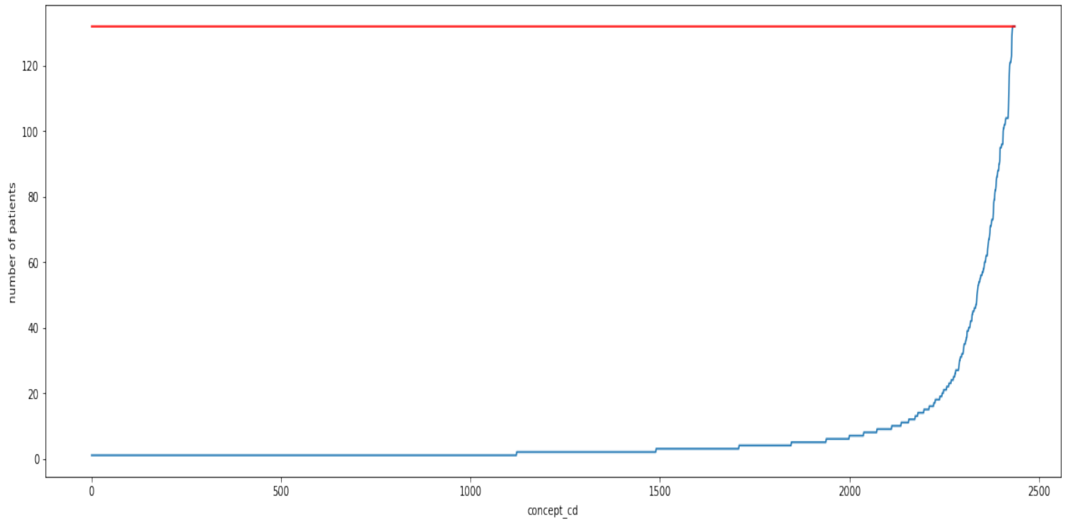

Fig. 2. Histogram with dummies

This would guarantee the maximum indistinguishability, but it would also imply a ratio ~17.
After defining the dissimilarity between all the ontology codes, applying HAC yields to the following result:

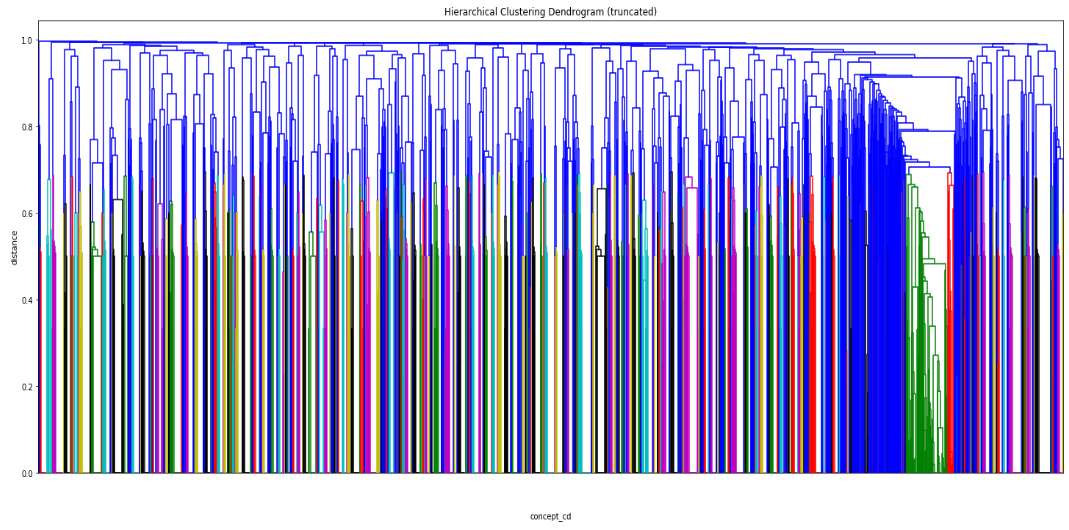

Fig. 3. Hierarchical Agglomerative Clustering

The x axis represents the different ontology codes, the y axis represents the distance between codes (or clusters). The dendrogram clearly shows that in general the codes are poorly correlated (the only exception is the green cluster on the right of the graph). As result, it does not really matter at which distance the dendrogram is cut for taking the clusters: there will be several small clusters with very few codes and just one with a reasonable number of codes.
Therefore, the following step has to make sure that a proper number of clusters were generated. If we look at the histogram 1, we can easily deduce that, as a rule of thumb, the frequency of the codes can be considered as a reasonable criterion for grouping them. Therefore, it makes sense to create further clusters, but by considering the frequency of codes as features. This leads to clusters in which all the codes together are either very correlated or their frequencies are similar, like in the following graph (ratio ~2.8):

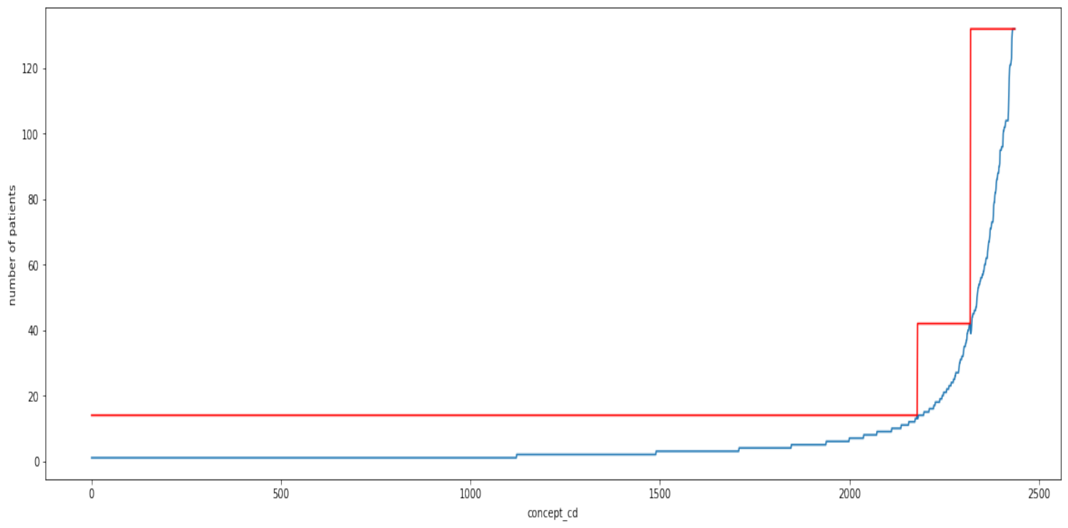

Fig. 4. Histogram Clustered

One obvious questions could relate to whether the same result could have been achieved by applying a clustering technique that directly considers only the frequency of codes. While the result would be pretty similar to the above graph, it would be impossible to ensure that all the highly correlated codes (even the ones that do not have similar frequency) would end up in the same cluster. Therefore, this is the benefit of using HAC as first step of the clustering phase.
While the aforementioned approach ensures that the ratio of generated dummies is reduced significantly, the size of the clusters is not really controlled. Since a minimum anonymity set must be guaranteed, the current approach is to take uncorrelated codes from the biggest cluster and put them into the others. This is shown in the below exhibit (where we set 400 as minimum anonymity set and get a ratio ~5):

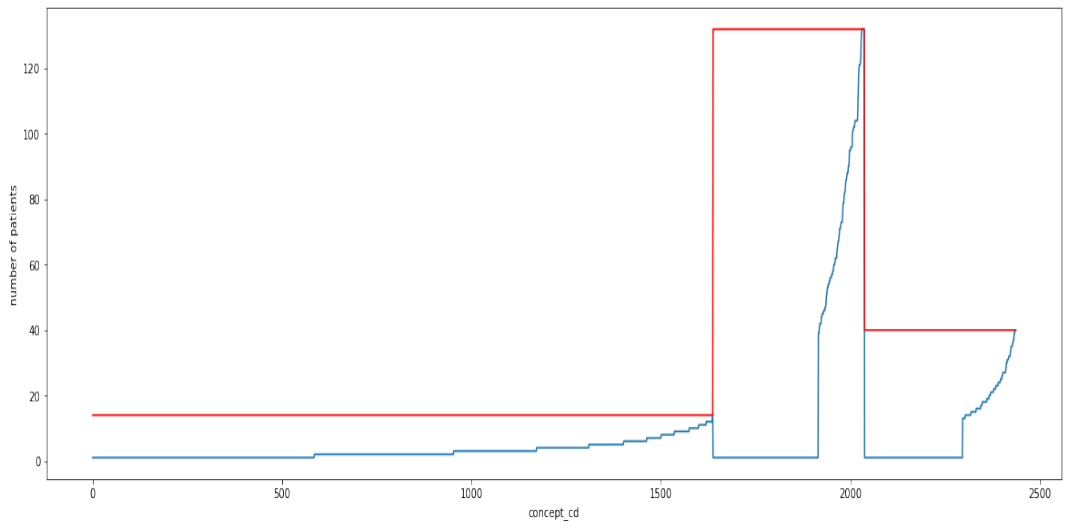

Fig. 5. Histogram Clustered

This approach may be dataset-dependent and it might need revision in the future.

3. DUMMY GENERATION

The second phase involves the actual dummies generation and it is be based on a water-fill algorithm whose goal will be to ensure that all codes within each bucket will have the same frequency. This is done by adding fake patients (dummies) with proper codes, in such way that the codes' occurrences increases lead to have ”flattened” clusters (i.e. like the red line shown in figure 5).

3.1. Requirements

As mentioned, the goal of the dummy generation phase is to ensure that all codes within the same cluster have the same frequency. This is achieved by creating fake patients, from the **real ones**. This means that each generated patient will have the same weight (number of codes associated) as a real patient. One constraint is that there cannot be two patients with the same combination of codes **within the same cluster** – this is stricter than simple inequality of codes between patients.

3.2. Approach

The followed approach consists in generating patients in order to fill the clusters of the histogram. It is clear from the requirements (3.1) that a naive water filling solution would not work: each new patient must be different from the existing ones (real and previously generated ones).
Since the goal is to fill all the clusters, whenever a patient is generated we must take into account which cluster we want to fill. In particular, when filling a specific cluster i we consider the number of codes (let's say k) that need more occurrences in order to have make the histogram look flat. Then, the goal would be to find a patient that has k codes, shuffle them, and add the new patient to the existing ones.
This may lead to some problems:

- After the shuffling, the patient may have the same combination of codes as an existing one
- There may be no patient with k codes in the cluster i

The first issue does not occur very often: the probability of having a collision within the same cluster is quite low if the minimum anonymity set is sufficiently high. In case this occurs, a fixing routine is called: it focuses on slightly changing the duplicate combination of codes in each cluster until it is different from the ones of all the other patients.
The second one is addressed by simply taking a patient that has less (or more) codes than necessary. The first alternative (taking a patient with less codes than necessary) is preferred over the second one. In fact, whenever a patient with *k'* codes is duplicated (with $k' > k$), there are k' - k codes that end up increasing the occurrences for codes that do not need more frequency. This causes histogram's maximum value (for cluster i) to increase. However, experiments have shown that most of the times patients with a smaller number of needed codes are available.
In this case, the codes needing more frequency are **randomly shuffled** in order to ensure that a subset of them is associated to the new dummy patient. It should be noticed that, although the algorithm focuses on filling one cluster at a time, the same approach is followed in all the other clusters as well. It doesn't matter whether the patient chosen for duplication has more/less codes in those clusters than the needed ones: it will still try to get the best combination of codes in other to fill them. This is done in order to avoid (as much as possible) the unpleasant situation in which filling a cluster leads the other ones to overflow. Several time, this results in having the procedure to fill one cluster filling completely the others as well (or getting close to have them fully filled).
One last remark is that in order to avoid recognition between real patients and fake ones, all the real ones must be duplicated. This is done at the beginning, before the water filling procedure.

3.3. Results

These experiments were performed by concatenating the modifier code with concept code. This operation led to have more concept codes (around 4000) than in the previous graphs. Figure 6 compares the original histogram (blue line) with the theoretical (red) and real (green) histograms after dummy generation. The first thing that can be noticed is that the histogram obtained after the generation of dummies is "higher" than the theoretical one. This results in a higher ratio in the former case. Also, in order to make the termination of the algorithm easier, we consider the histogram filled even if its frequencies are below (at most by 1) the target one.
Table 7 summarizes some experiments that were performed. As a rule of thumb, we can see that increasing the minimum allowed anonymity set and decreasing the number of clusters lead to have a higher ratio. However, it also depends on how the clustering is performed and how the dummies are generated: this means that the obtained ratio is not fully deterministic.

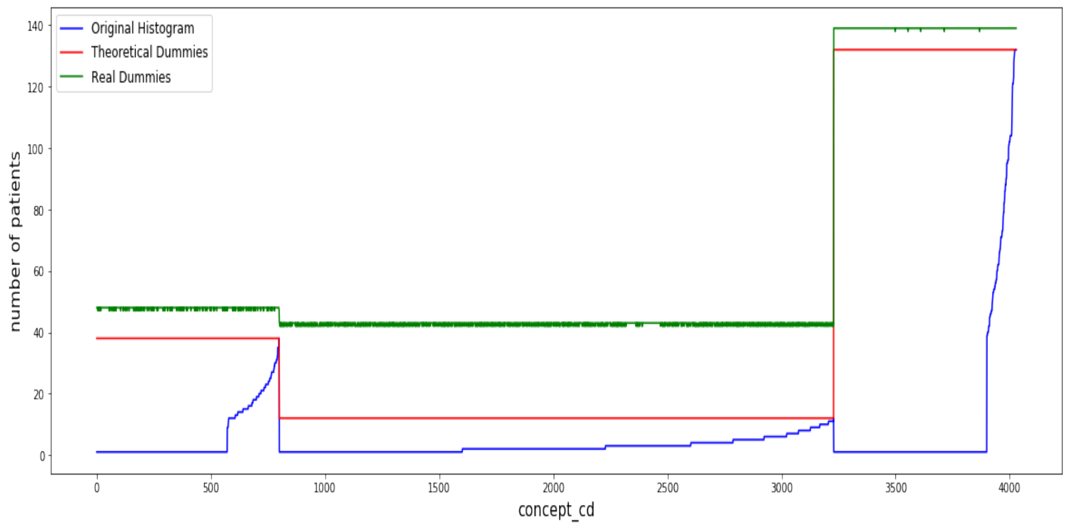

Fig. 6. Histogram Clustered

Fig. 7. Experiments

Minimum Anonymity Set	Clusters	Clusters' filling order	Theoretical ratio	Real ratio
600	2	Ascending	7.6	8.9
600	2	Descending	7.6	8.2
600	3	Ascending	6	8.9
600	3	Descending	6	9.1
800	2	Ascending	8.5	11.2
800	2	Descending	8.5	10.1
800	3	Ascending	7.2	12
800	3	Descending	7.33	10.9
1000	2	Ascending	9.5	13.9
1000	2	Descending	9.5	12.6
1000	3	Ascending	8.6	15
1000	3	Descending	8.5	13.9

4. FUTURE WORK

The proposed approach proved to be a valid solution for the considered problem. In particular, it allowed us to define some meaningful clusters – by making sure that highly correlated codes would end up in the same cluster and that the constraints that relate to the size of the clusters are met – and to generate dummy patients. It should be noticed, however, that the results are highly influenced by the dataset: the same approach should definitely be tested with other data as well. Furthermore, if bigger datasets are used, a distributed and parallelized implementation should be seriously taken into consideration – especially for the clustering phase.

Although the results are meaningful, there is space for further experimentation and improvement. For instance, if several dummies are generated, it may happen that the height of a cluster exceeds the ideal height of another cluster (like in figure 6). In this case it might make sense to just merge those clusters. One criterion for detecting this case could be a theoretical analysis, based on the theoretical filled histogram. In particular, if we see that the number of patients that are needed in order to fill a cluster cause the height of another cluster to surpass the height of a third cluster, then we can merge those clusters. This happens in figure 6: even if we ignore the green line, it can already be seen that by water filling the third cluster we would cause the height of the second cluster to surpass the height of the first cluster.