Clustering the Normalized Compression Distance for Virus Data

Kimihito Ito[1], Thomas Zeugmann[2]* and Yu Zhu[2]

[1] Research Center for Zoonosis Control
Hokkaido University, N-20, W-10 Kita-ku, Sapporo 001-0020, Japan
`itok@czc.hokudai.ac.jp`
[2] Division of Computer Science
Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan
{`thomas,yuar`}`@mx-alg.ist.hokudai.ac.jp`

Abstract. The present paper analyzes the usefulness of the normalized compression distance for the problem to cluster the HA sequences of virus data for the HA gene in dependence on the available compressors. Using the CompLearn Toolkit, the built-in compressors `zlib` and `bzip` are compared.
Moreover, a comparison is made with respect to hierarchical and spectral clustering. For the hierarchical clustering, `hclust` from the R package is used, and the spectral clustering is done via the kLine algorithm proposed by Fischer and Poland (2004).
Our results are very promising and show that one can obtain an (almost) perfect clustering. It turned out that the `zlib` compressor allowed for better results than the `bzip` compressor and, if all data are concerned, then hierarchical clustering is a bit better than spectral clustering via kLines.

1 Introduction

The similarity between objects is a fundamental notion in everyday life. It is also fundamental to many data mining and machine learning algorithms, and, in particular to clustering algorithms. Often the similarity between objects is measured by a domain-specific distance measure based on features of the objects. For example, the distance between pieces of music can be measured by using features like rhythm, pitch, or melody, i.e., features that do not make sense in any other domain. To develop such methods one needs special knowledge about the application domain for extracting the relevant features beforehand. Such an approach does not only cause difficulties, but includes a certain danger or risk of being biased.

If one is pursuing the approach to design data mining algorithms based on domain knowledge, then the resulting algorithms tend to have many parameters. By using these parameters, one can then control the algorithms' sensitivity to certain features. Determining how relevant particular features are is often difficult and may require a certain amount of guessing. Expressing this differently, one has to tune the algorithms which is requiring domain knowledge and a larger amount of experience. Furthermore, it may be expensive, error prune and time consuming to arrive at a suitable tuning.

However, as a radically different approach, the paradigm of parameter-free data mining has emerged (cf. Keogh *et al.* [11]). The main idea of parameter-free data mining is the design of algorithms that have no parameters and that are universally applicable in all areas.

The problem is whether or not such an approach can be realized at all. It is only natural to ask how an algorithm can perform well if it is not based on extracting the important features of the data and if we are not allowed to adjust its parameters until it is doing the right thing. As expressed by Vitányi *et al.* [21], *if we a priori know the features, how to extract them, and how to combine them into exactly the distance measure we want, we should do just that. For example, if we have a list of cars with their color, motor rating, etc. and want to cluster them by color, we can easily do that in a straightforward way.*

So the approach of parameter-free data mining is aiming at scenarios where we are not interested in a certain similarity measure but in the similarity between the objects themselves. The most promising approach to this paradigm uses Kolmogorov complexity theory [13] as its basis.

The key ingredient to this approach is the so-called *normalized information distance* (NID) which was developed by various researchers during the past decade in a series of steps (cf., e.g., [4, 12, 9]).

More formally the *normalized information distance* between two strings $\mathtt{x}$ and $\mathtt{y}$ is defined as

$$NID(\mathtt{x},\mathtt{y}) = \frac{\max\{\mathsf{K}(\mathtt{x}|\mathtt{y}),\ \mathsf{K}(\mathtt{y}|\mathtt{x})\}}{\max\{\mathsf{K}(\mathtt{x}),\ \mathsf{K}(\mathtt{y})\}}\ , \tag{1}$$

* Supported by MEXT Grant-in-Aid for Scientific Research on Priority Areas under Grant No. 21013001.

where $K(x|y)$ is the length of the shortest program that outputs x on input y, and $K(x)$ is the length of the shortest program that outputs x on the empty input. It is beyond the scope of the present paper to discuss the technical details of the definition of the NID. We refer the reader to Vitányi *et al.* [21].

The NID has nice theoretical properties, the most important of which is universality. The NID is called *universal*, since it accounts for the dominant difference between two objects (cf. Li *et al.* [12] and Vitányi *et al.* [21] and the references therein).

In a sense, the NID captures all computational ways in which the features needed in the traditional approach could be defined. Since its definition involves the Kolmogorov complexity $K(\cdot)$, the NID cannot be computed. Therefore, to apply this idea to real-world data mining tasks, standard compression algorithms, such as `gzip`, `bzip2`, or `PPMZ`, have been used as approximations of the Kolmogorov complexity. This yields the *normalized compression distance* (NCD) as approximation of the NID (cf. Definition 1).

In a typical data mining scenario we are given some objects as input. The pairwise NCDs for all objects in question form a distance matrix. This matrix can be processed further until finally standard algorithms, e. g., clustering algorithms can be applied. This has been done in a variety of typical data mining scenarios with remarkable success. Works of literature and music have been clustered according to genre or author; evolutionary trees of mammals have been derived from their mitochondrial genome; language trees have been derived from several linguistic corpora (cf., e.g., [9, 11, 6, 7, 3]).

As far as virus data are concerned, Cilibrasi and Vitányi [8] used the SARS TOR2 draft genome assembly 120403 from Canada's Michael Smith Genome Sciences Centre and compared it to other viruses by using the NCD. They used the `bzip2` compressor and applied their quartet tree heuristic for hierarchical clustering. The resulting ternary tree showed relations very similar to those shown in the definitive tree based on medical-macrobiological genomics analysis which was obtained later (see [8] for details).

The main goal of the present paper is a detailed analysis of the general method outlined above in the domain of influenza viruses. More specifically, we are interested in learning whether or not specific gene data for the hemagglutinin of influenza viruses are *correctly* classifiable by using the concept of the NCD. For this purpose we have chosen a set of 106 gene sequences from the National Center for Biotechnology Information for which the correct classification of the hemagglutinin is known. As explained in Section 3, there are 16 subtypes commonly called H1, . . . , H16. For these 106 gene sequences (or subsets thereof) we then compute the NCD by using the CompLearn Toolkit (cf. [5]) as done in [8].

This computation returns a symmetric matrix D such that d_{ij} is the NCD between the data entries i and j (henceforth called distance matrix). Furthermore, we study the influence of the compressor chosen and restrict ourselves here to the `zlib` and `bzip2` compressors which are the standard two built-in compressors for the CompLearn Toolkit.

The next step is the clustering. Here of course the variety of possible algorithms is large. Note that the CompLearn Toolkit contains also an implementation of quartet tree heuristic for hierarchical clustering. However, this heuristic is computationally quite expensive and does currently not allow to handle a matrix of dimension 106×106. Therefore, we have decided to try the *hierarchical clustering* algorithm from the R package (called `hclust`) with the average option. In this way we obtain a rooted tree showing the relations among the input data.

The second clustering algorithm used is *spectral clustering* via kLines (cf. Fischer and Poland [10]). We have successfully applied this method before (cf. [19, 18]) in settings where the NID is approximated by the so-called Google Distance. It should be noted that spectral clustering generally requires the transformation of the distance matrix into an adjacency matrix of pairwise similarities (henceforth called similarity matrix). The clustering is then done by analyzing its spectrum.

The results obtained are generally very promising. Quite often, we obtained a *perfect* clustering independently of the method used. On the other hand, when including all data or a rather large subset thereof, the clustering obtained is not perfect but the number of errors made is still sufficiently small to make the results interesting. Without going into details here, it can be said that the `zlib` compressor seems more suitable in this setting than the `bzip2` compressor (see Subsection 3.2 for details).

2 Background and Theory

As explained in the Introduction, the theoretical basis for computing the distance matrix is deeply based in Kolmogorov complexity theory. In the following we assume the definition of the NID as shown in Equation (1). The definition of the NID depends on the function K which is *uncomputable.* Thus, the NID is *uncomputable*, too.

Using a real-word compressor, one can approximate the NID by the NCD (cf. Definition 1). Again, we omit details and refer the reader to [21].

Definition 1. *The* normalized compression distance *between two strings* x *and* y *is defined as*

$$NCD(x,y) = \frac{C(xy) - \min\{C(x),\ C(y)\}}{\max\{C(x),\ C(y)\}}\ ,$$

where C *is any given data compressor.*

Common data compressors are `gzip`, `bzip2`, `zlib`, etc. Note that the compressor C has to be computable and *normal* in order to make the NCD a useful approximation. This can be stated as follows.

Definition 2 ([21]). *A compressor* C *is said to be* normal *if it satisfies the following axioms for all strings* x, y, z *and the empty string* λ.

(1) $C(xx) = C(x)$ *and* $C(\lambda) = 0$*;* (identity)
(2) $C(xy) \geqslant C(x)$*;* (monotonicity)
(3) $C(xy) = C(yx)$*;* (symmetry)
(4) $C(xy) + C(z) \leqslant C(xz) + C(yz)$*;* (distributivity)

up to an additive $O(\log n)$ *term, with* n *the maximal binary length of a string involved in the (in)equality concerned.*

These axioms are in various degrees satisfied by good real-world compressors like `bzip2`, `PPMZ` and `gzip`, where the latter did not perform so well, as informal experiments have shown (cf. [9]). Also note that in all cases the compressor-specific window or block size determines the maximum usable length of the arguments. As a matter of fact, for our data these axioms seem to be fulfilled.

For our investigations we used the built-in compressors `bzip2` and `zlib` and the `ncd` function from the CompLearn Toolkit (cf. [5]). After having done this step, we have a distance matrix $D = \left(d^{\text{ncd}}(x,y)\right)_{x,y\in X}$, where $X = (x_1, \ldots, x_n)$ is the relevant data list.

Next, we turn our attention to clustering. First, we shortly outline the hierarchical clustering as provided by the R package, i.e., by the program `hclust` (cf. [2]). Input is the $(n \times n)$ distance matrix D. The program uses a measure of dissimilarity for the objects to be clustered. Initially, each object is assigned to its own cluster and the program proceeds iteratively. In each iteration the two most similar clusters are joint, and the process is repeated until only a single cluster is left. Furthermore, in every iteration the distances between clusters are recomputed by using the Lance–Williams dissimilarity update formula for the particular method used.

The methods differ in the way in which the distances between clusters are recomputed. Provided are the *complete linkage method*, the *single linkage method*, and the *average linkage clustering*. In the first case, the distance between any two clusters is equal to the greatest similarity from any member of one cluster to any member of the other cluster. This method works well for compact clusters but causes sensitivity to outliers. The second method pays attention solely to the area where the two clusters come closest to one another. The more distant parts of the clusters and the overall structure of the clusters is not taken into account. If the total number of clusters is large, a messy clustering may result.

The *average linkage clustering* defines the distance between any two clusters to be the average of distances between all pairs of objects from any member of one cluster to any member of the other cluster. As a result, the average pairwise distance within the newly formed cluster, is minimum.

Heuristically, the average linkage clustering should give the best results in our setting, and thus we have chosen it (see also Manning *et al.* [14] for a thorough exposition). Note that for hierarchical clustering the number k of clusters does *not* to be known in advance.

Next, the *spectral clustering* algorithm used is shortly explained. Spectral clustering is an increasingly popular method for analyzing and clustering data by using only the matrix of pairwise similarities. It was invented more than 30 years ago for partitioning graphs (cf., e.g., Spielman and Teng [20] for a brief history and Luxburg [22] for a tutorial). Formally, spectral clustering can be related to approximating the normalized min-cut of the graph defined by the adjacency matrix of pairwise similarities [24]. Finding the exactly minimizing cut is an NP-hard problem.

The transformation of the distance matrix into a similarity matrix is done by using a suitable kernel function. In our experiments we have used the Gaussian kernel function, i.e.,

$$k(x,y) = \left(\exp(-\frac{1}{2}d(x,y)^2/(2\cdot\sigma^2))\right)\ , \tag{2}$$

where σ is the kernel width. As pointed out by Perona and Freeman [17], there is nothing magical with this function. Moreover, it is most commonly used. An advantage of using the Gaussian kernel function is that the resulting similarity matrix is positive definite.

So, the remaining problem is a suitable choice for σ. Unfortunately, the performance of spectral clustering heavily depends on this σ. In the experiments, we compute the mean value of the entries of the distance matrix D and then set $\sigma = \mathtt{mean}(D)/\sqrt{2}$. In this way, the kernel is most sensitive around $\mathtt{mean}(D)$. Though we are not aware of a theoretical result supporting this choice, it worked remarkably well and further studies are needed to explore the properties of this choice.

The final spectral clustering algorithm for a known number of clusters k is stated below.

Algorithm *Spectral Clustering of a Data List*
Input: data list $X = (x_1, x_2, \dots, x_n)$, number of clusters k
Output: clustering $c \in \{1 \dots k\}^n$

```
1. for $x, y \in X$, compute the distance matrix $D = \left(d^{\text{ncd}}(x, y)\right)_{x,y \in X}$
2. compute $\sigma = \mathtt{mean}(D)/\sqrt{2}$
3. compute the similarity matrix $A = \left(\exp(-\frac{1}{2} d(x, y)^2/(2 \cdot \sigma^2))\right)$
4. compute the Laplacian $L = S^{-\frac{1}{2}} A S^{-\frac{1}{2}}$, where $S_{ii} = \sum_j A_{ij}$ and $S_{ij} = 0$ for $i \neq j$
5. compute top $k$ eigenvectors $V \in \mathbb{R}^{n \times k}$
6. cluster $V$ using kLines [10]
```

3 Experiments and Results

In this section we describe the data used, the experiments performed and the results obtained.

3.1 Influenza Viruses - The Data Set

We shortly describe the data set used. For any relevant background concerning the biological aspects of the influenza viruses we refer the reader to Palese and Shaw [16] and Wright *et al.* [23].

Influenza viruses were probably a major cause of morbidity and mortality world wide. Large segments of the human population are affected every year. The family of *Orthomyxoviridae* is defined by viruses that have a negative-sense, single-stranded, and segmented RNA genome. There are five different genera in the family of *Orthomyxoviridae*: the influenza viruses A, B and C; *Thogotovirus*; and *Isavirus*. Influenza A viruses have a complex structure and possess a lipid membrane derived from the host cell (cf. Figure 1).

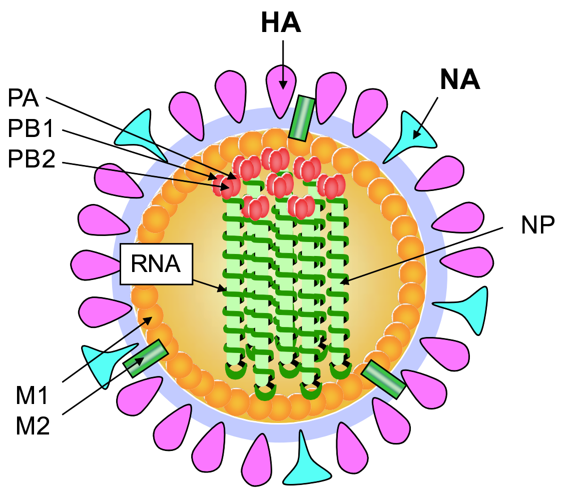

Fig. 1. Influenza A virus

Biologists classify influenza A viruses primarily by their hemagglutinin (HA) subtypes and neuraminidase (NA) subtypes. So far, 16 subtypes of HA are known and commonly denoted by H1, ..., H16. In addition to these HA types, biologists distinguish 9 NA subtypes denoted by N1, ..., N9.

Influenza A viruses of all 16 hemagglutinin (H1-H16) and 9 neuraminidase (N1-N9) subtypes are maintained in their nature host, i.e., the duck. Of these duck viruses, H1N1, H2N2 and H3N2 subtypes jumped into human population, and caused three pandemics in the last century. Therefore, in the experiments performed we have exclusively selected data of influenza viruses that have been obtained from viruses hosted by the duck.

The complete genome of these influenza viruses has 8 segmented-genes. Of these 8 genes, here we are only interested in their HA gene, since HA is the major target of antibodies that neutralize viral infectivity, and responsible for binding the virus to the cell it infects. The corresponding gene is found on segment 4.

Each datum consists of a sequence of roughly 1800 letters from the alphabet {A, T, G, C}, e.g., looking such as AAAAGCAGGGGAATTTCACAATTAAACAAAAT...TGTATATAATTAGCAAA. These gene sequences are publicly available from the National Center for Biotechnology Information (NCBI) which has one of the largest collections of such sequences (cf. [15]).

When analyzed by biologists the definite method to determine the correct HA subtype is based on the antiserum that prevent the docking of the virus. Sometimes biologists also compare the actual sequence to already analyzed sequences and produce a guess based on the Hamming distance of the new sequence to the analyzed ones.

As explained in the Introduction, the primary goal of the investigations undertaken is to cluster the sequences correctly with respect to their HA subtype. In order to achieve this goal with collected from each subtype up to 8 examples. The reason for choosing at most 8 sequences from each type has been caused by their availability. While for some subtypes there are many sequences, there are also subtypes for which only very few sequences are available. The extreme case is the subtype H16 for which only one sequence is in the data base. Table 2 shows the number of sequences chosen.

H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12	H13	H14	H15	H16
8	8	8	8	8	8	8	7	8	8	8	8	2	4	4	1

Fig. 2. Number of sequences for each subtype

It should be noted that most of these sequences are marked as `complete cds`, but some are also marked as `partial cds` by the NCBI. For a complete list of the data description we refer the reader to `http://www-alg.ist.hokudai.ac.jp/datasets.html` .

For the ease of presentation, below we use the following abbreviation for the data entries. Instead of giving the full description, e.g.,
>gi|113531192|gb|AB271117| /Avian/4 (HA)/H10N1/Hong Kong/1980/// Influenza A virus (A/duck/Hong Kong/938/80(H10N1)) HA gene for hemagglutinin, complete cds.

we refer to this datum as H10N1AB271117 for short.

Among the available files, there were two files containing only a very short partial sequence of the gene, i.e., H7N1AM157391 and H10N4AM922160 (483 and 80 letters, respectively). So, we did not consider these two files, since they do not seem to contain enough information.

3.2 Results

All experiments have been performed under SuSE Linux. As already mentioned, for the hierarchical clustering we used the open source R package (cf. [2]).

The Algorithm *Spectral Clustering of a Data List* from Section 2 has been realized by performing Step 1 via the CompLearn function `ncd` (cf. [5]). Steps 2 through 6 have been implemented in `GNU Octave`, version 2.1.72 (cf. [1]). It should be noted that `ncd` assigns 0.000000 to all elements on the main diagonal of the distance matrix (Version 1.1.5).

By performing our experiments we aimed to answer the following questions. First, does the NCD provide enough information to obtain a correct clustering for the virus data? Second, does the rather large number of clusters (recall that we 16 HA types) cause any problems? Third, do the answers to the first and second question depend on the compressor and clustering, respectively, chosen?

To get started and for the sake of comparison, we used the subset containing all data belonging to H1, H2, and H3, i.e., a total of 24 sequences (cf. Figure 2).

Using the `maketree` program from the CompLearn Toolkit, we get the following clustering (cf. Figures 3 and 4). As Figures 3 and 4 show, the data are clearly and correctly separated into three clusters.

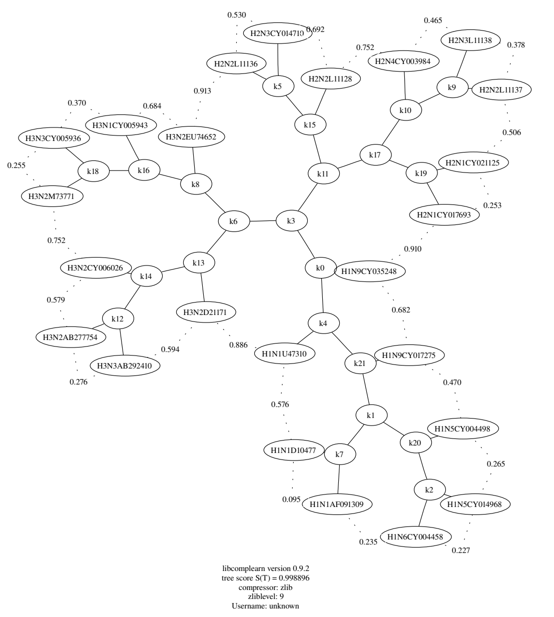

Fig. 3. Classification of HA sequences; compr.: `zlib`

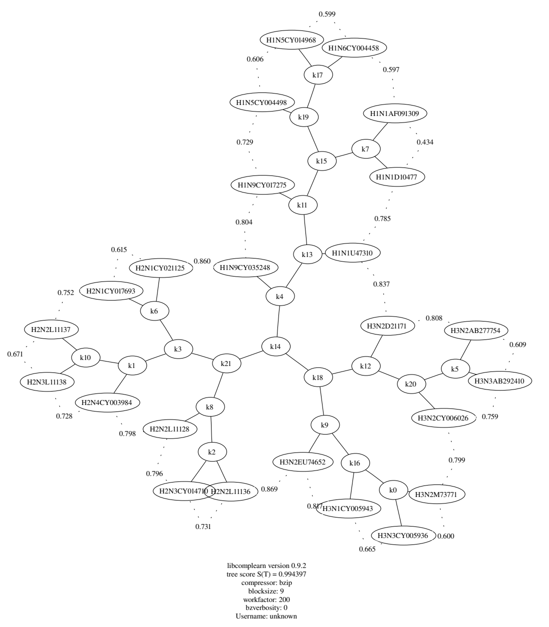

Fig. 4. Classification of HA sequences; compr.: `bzip`

However, the intra-cluster dissimilarities clearly differ from inter-cluster dissimilarities in Figure 3, i.e., for the `zlib` compressor, while there is no such clear difference for the `bzip` compressor (cf. Figure 4).

Using `hclust` we obtained the trees shown in Figure 5 and 6 for the matrix D computed for the compressor `zlib` and `bzip`, respectively.

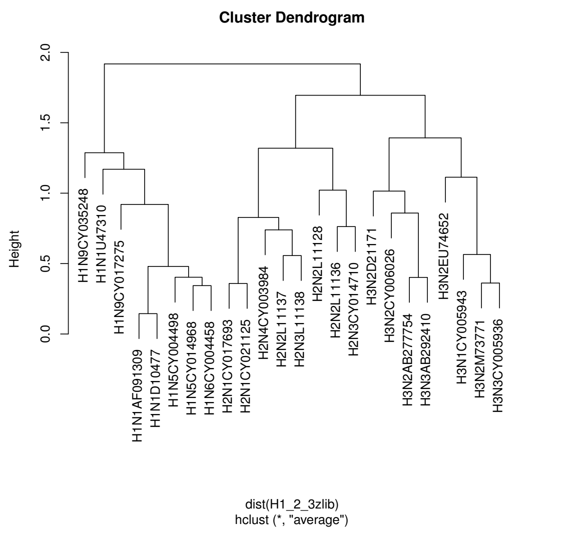

Fig. 5. Clustering all HA sequences for H1 through H3 via `hclust`; compr.: `zlib`

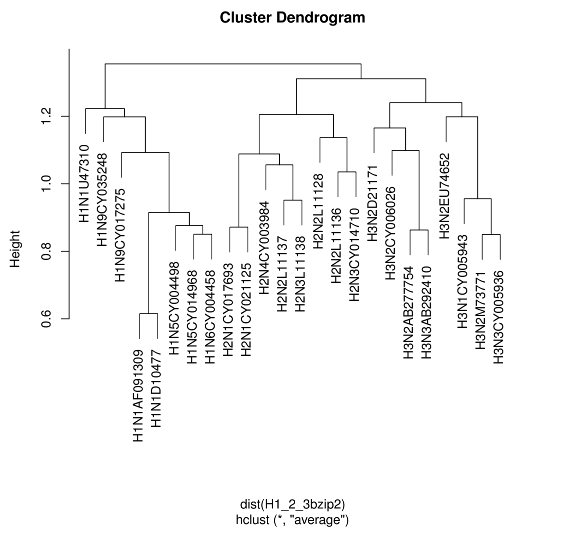

Fig. 6. Clustering all HA sequences for H1 through H3 via `hclust`; compr.: `bzip`

After having computed the matrix D, we get the following order of the data

```
H2N4CY003984, H3N1CY005943, H3N2AB277754, H1N9CY017275, H1N9CY035248, H3N3CY005936,
H2N2L11128, H2N2L11136, H2N2L11137, H2N1CY017693, H2N1CY021125, H2N3L11138,
H1N6CY004458, H1N1D10477, H2N3CY014710, H3N2EU74652, H3N2CY006026, H1N1AF091309,
H1N1U47310, H3N3AB292410, H3N2D21171, H3N2M73771, H1N5CY004498, H1N5CY014968
```

Since spectral clustering is a hard clustering method, it has to return for each data entry just one class label. Assigning canonically the clusters 1, 2, and 3 to the HA subtypes, we therefore should get the sequence

2 3 3 1 1 3 2 2 2 2 2 2 2 1 1 2 3 3 1 1 3 3 3 1 1

which was indeed returned for both compressors. Note that $\sigma = 0.56078$ and $\sigma = 0.57329$ for the `zlib` and `bzip` compressor, respectively.

Next, we tried all HA sequences for H1 through H8 and from H9 through H16. The reason for this partition has been caused by the different number of sequences available. Recall that there are only two sequences for H13 and only one sequence for H16 (cf. Figure 2).

For H1 through H8 the hierarchical clustering was error free for the `zlib` compressor but not for `bzip` compressor (1 error) (see Figures 7 and 8 in the Appendix). Interestingly, for H9 through H16 the tree obtained for the `zlib` compressor contains 4 errors, while the one obtained for `bzip` compressor has only one error.

Our spectral clustering algorithm returned a perfect clustering for all HA sequences for H1 through H8 for both compressors. On the other hand, for all sequences from H9 through H16 the results differed with respect to the compressors.

For the `zlib` compressor we obtained 5 errors and for `bzip` the number of errors was 7 when using for σ the mean as described above. However, it is well-known that spectral clustering is quite sensitive to the kernel width σ. So, we also tried to vary it a bit around the mean by rounding it to two decimal digits and then changing the second one. For `zlib` the mean was 0.60873 and after two variations we found $\sigma = 0.59$ which resulted in just one error, i.e., H16 was classified as H13. For the `bzip` compressor such an improvement could not be obtained.

As a possible explanation we conjecture that one needs a certain minimum of available sequences in order to arrive at a correct spectral clustering. Trying all HA sequences for H1 through H12 kind of confirmed this conjecture, since we again obtained a perfect spectral clustering for both compressors.

For the hierarchical clustering, the tree obtained for the `zlib` compressor is correct, but the the one obtained for the `bzip` compressor has one error. These trees are shown in the Appendix.

Finally, we tried all data. Again hierarchical clustering was best for the `zlib` compressor and showed only 2 errors. For the `bzip` compressor, we obtained 3 errors (see the Appendix for details). On the other hand, the best result we could obtain for spectral clustering had 5 errors (for both compressors). Below we show the clustering obtained for the `zlib` compressor for $\sigma = 0.63$, where c0 is the desired classification and sp the one returned from the spectral clustering algorithm (partitioned into four groups).

c0 =	7	7	14	2	11	12	12	3	7	10	10	5	9	9	9	3	1	1	9	11	11	5	3	7	5	2	2	2	4	10
sp =	7	7	14	2	11	12	12	3	7	10	10	5	9	9	9	3	1	1	9	11	11	5	3	7	5	2	2	2	4	10
c0 =	5	8	12	2	2	4	4	4	11	9	10	2	6	6	6	5	1	1	4	10	7	4	8	15	2	9	9	16	10	14
sp =	5	8	12	2	2	4	4	4	11	9	10	2	13	13	6	5	1	1	4	10	7	4	8	15	2	9	9	3	10	14
c0 =	14	7	7	6	14	7	8	8	12	12	11	15	3	15	5	11	3	1	1	8	4	3	3	6	12	10	4	5	3	6
sp =	14	7	7	6	14	7	8	8	12	12	11	15	3	15	5	11	3	1	1	8	4	3	3	6	12	10	4	5	3	6
c0 =	13	13	12	1	1	11	12	8	11	10	5	9	15	8	6	6														
sp =	13	13	12	1	1	11	12	8	11	10	5	9	15	8	13	13														

So, the errors occur at positions 43, 44, 58, 105, and 106 and affect H6 which is four times assigned to H13 and one time H16 which got in the H3 cluster. We omit further details due to the lack of space.

Note that one can also compute the sum square error (s.s.e.) of all eigenvalues w.r.t. their means in order to determine quite reliably from the eigenvalues of the Laplacian the number k of clusters.

4 Conclusions

The usefulness of the normalized compression distance for clustering the HA type of virus data for the HA gene for it (segment 4) has been demonstrated. Though we just used the built-in compressors `zlib` and `bzip` the results are (almost) correct when clustering the resulting distance matrix for the whole data set with `hclust` or spectral clustering via kLines. What is also remarkable in this context is the robustness with respect to the completeness of the data. As mentioned above, some data contain only a partial cds but this did not influence the quality of the clustering as the results, e.g., H1N1U47310 and H3N2D21171 have only 1000 letters.

We have not reported the running time here, since it is still in the range of several seconds. Though the quartet tree algorithm by Cilibrasi and Vitányi [8] returns a high quality classification, it lacks scalability, since it tries to optimize a quality function, a task which is NP-hard. So, even for the small example including the 24 data for H1, H2, and H3 resulting in (24×24) distance matrix, it took hours to find the resulting (very good) clustering. In contrast, the clustering algorithms used in this study scale nicely at least up to the amount of data for which the distance matrix is efficiently computable, since they have almost the same running time as the `ncd` algorithm.

References

[1] GNU Octave. http://www.gnu.org/software/octave/.

[2] The R project for statistical computing. http://www.r-project.org/.

[3] D. Benedetto, E. Caglioti, and V. Loreto. Language trees and zipping. *Phys. Rev. Lett.*, 88(4):048702–1–048702–4, 2002.

[4] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitányi, and W. H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

[5] R. Cilibrasi. The CompLearn Toolkit, 2003-. http://www.complearn.org/.

[6] R. Cilibrasi and P. Vitányi. Automatic meaning discovery using Google. Manuscript, CWI, Amsterdam, 2006.

[7] R. Cilibrasi and P. Vitanyi. Similarity of objects and the meaning of words. In *Theory and Applications of Models of Computation, Third International Conference, TAMC 2006, Beijing, China, May 2006, Proceedings*, volume 3959 of *Lecture Notes in Computer Science*, pages 21–45, Berlin, 2006. Springer.

[8] R. Cilibrasi and P. M. Vitányi. A new quartet tree heuristic for hierarchical clustering. In D. V. Arnold, T. Jansen, M. D. Vose, and J. E. Rowe, editors, *Theory of Evolutionary Algorithms*, number 06061 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[9] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[10] I. Fischer and J. Poland. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA / USI-SUPSI, Manno, Switzerland, 2004.

[11] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM Press, 2004.

[12] M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[13] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer, 3rd edition, 2008.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[15] National Center for Biotechnology Information. Influenza Virus Resource, information, search and analysis. http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html.

[16] P. Palese and M. L. Shaw. Orthomyxoviridae: The viruses and their replication. In D. M. Knipe and P. M. H. et al., editors, *Fields' Virology*, pages 1647–1689. Lippincott Williams & Wilkins, Philadelphia, fifth edition, 2007.

[17] P. Perona and W. Freeman. A factorization approach to grouping. In H. Burkhardt and B. Neumann, editors, *5th European Conference on Computer Vision Freiburg, Germany, June, 2–6, 1998, Proceedings, Volume I*, Lecture Notes in Computer Science, pages 655–670. Springer, 1998.

[18] J. Poland and T. Zeugmann. Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. In *Discovery Science, 9th International Conference, DS 2006, Barcelona, Spain, October 2006, Proceedings*, volume 4265 of *Lecture Notes in Artificial Intelligence*, pages 197–208. Springer, 2006.

[19] J. Poland and T. Zeugmann. Clustering the google distance with eigenvectors and semidefinite programming. In *Knowledge Media Technologies, First International Core-to-Core Workshop*, volume 21 of *Diskussionsbeiträge, Institut für Medien und Kommunikationswisschaft*, pages 61–69. Technische Universität Ilmenau, 2006.

[20] D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the 37th Annual IEEE Conference on Foundations of Computer Science*, pages 96–105. IEEE Computer Society, 1996.

[21] P. M. B. Vitányi, F. J. Balbach, R. L. Cilibrasi, and M. Li. Normalized information distance. In *Information Theory and Statistical Learning*, pages 45–82. Springer, New York, 2008.

[22] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[23] P. F. Wright, G. Neumann, and Y. Kawaoka. Orthomyxoviruses. In D. M. Knipe and P. M. H. et al., editors, *Fields' Virology*, pages 1691–1740. Lippincott Williams & Wilkins, Philadelphia, fifth edition, 2007.

[24] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 313–319. IEEE Computer Society, 2003.

Appendix

Here we show the results obtained for the remaining data.

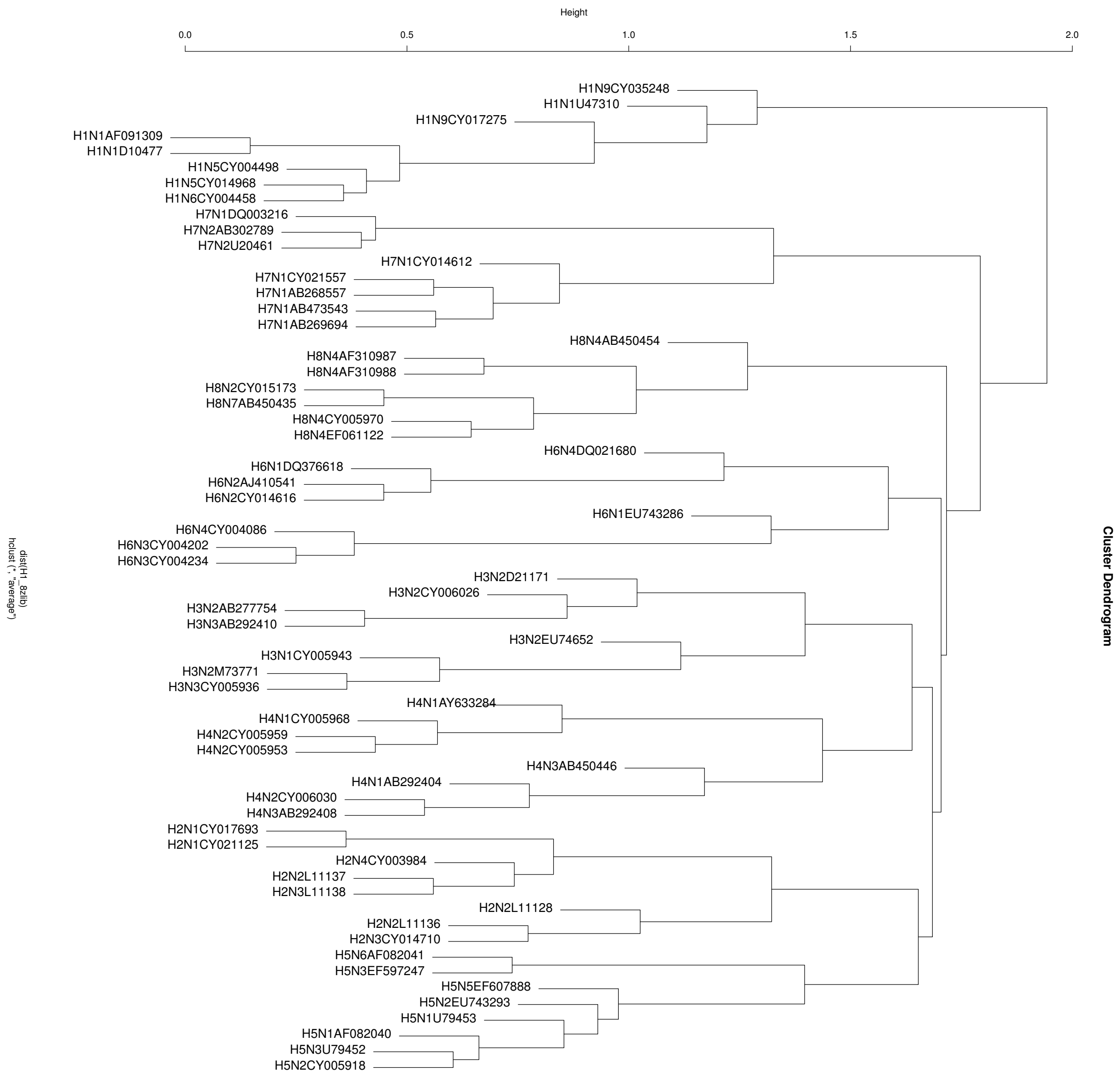

Fig. 7. Clustering of all HA sequences for H1 through H8 via `hclust`; compr.: `zlib`

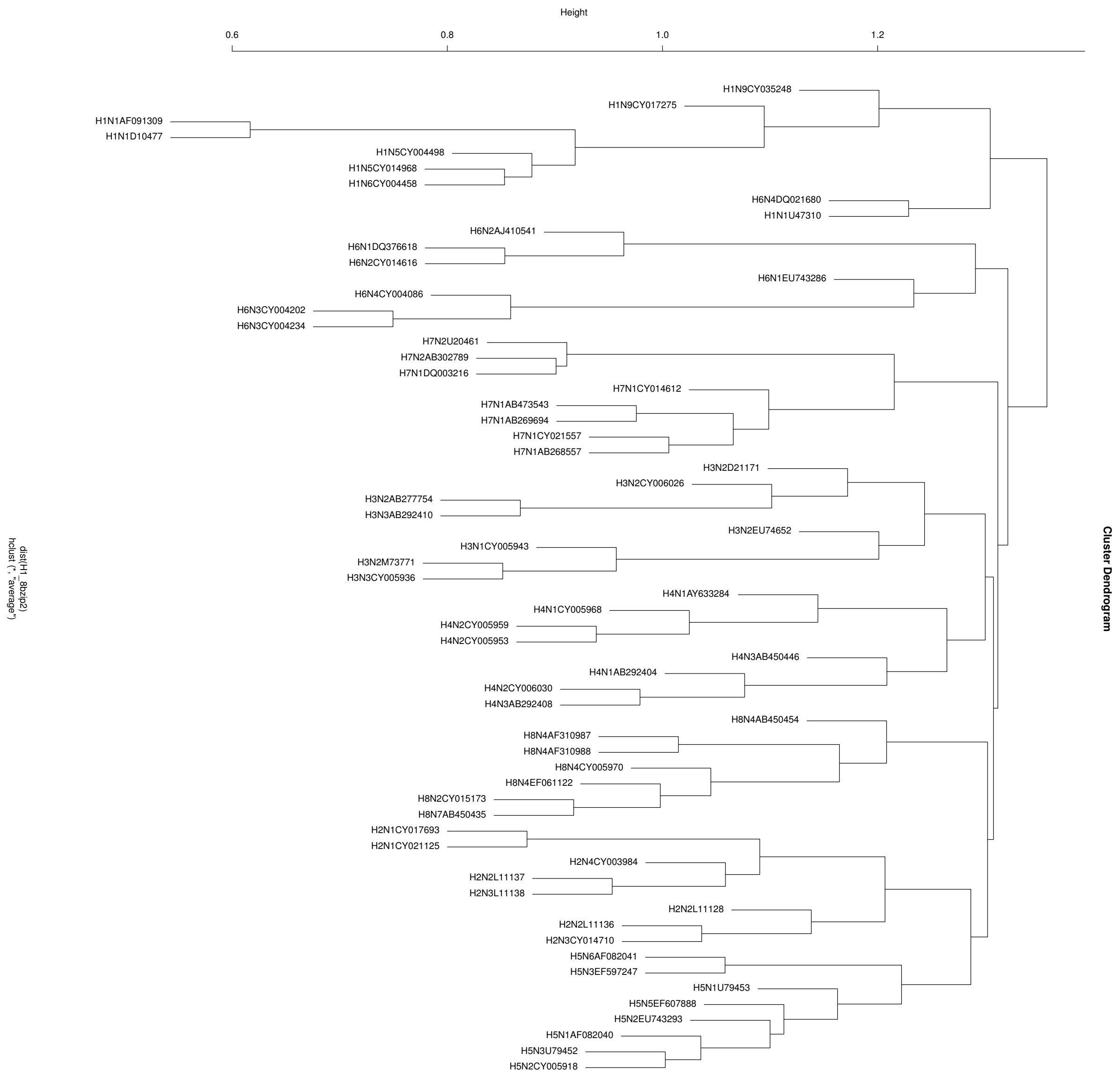

Fig. 8. Clustering of all HA sequences for H1 through H8 via `hclust`; compr.: `bzip`

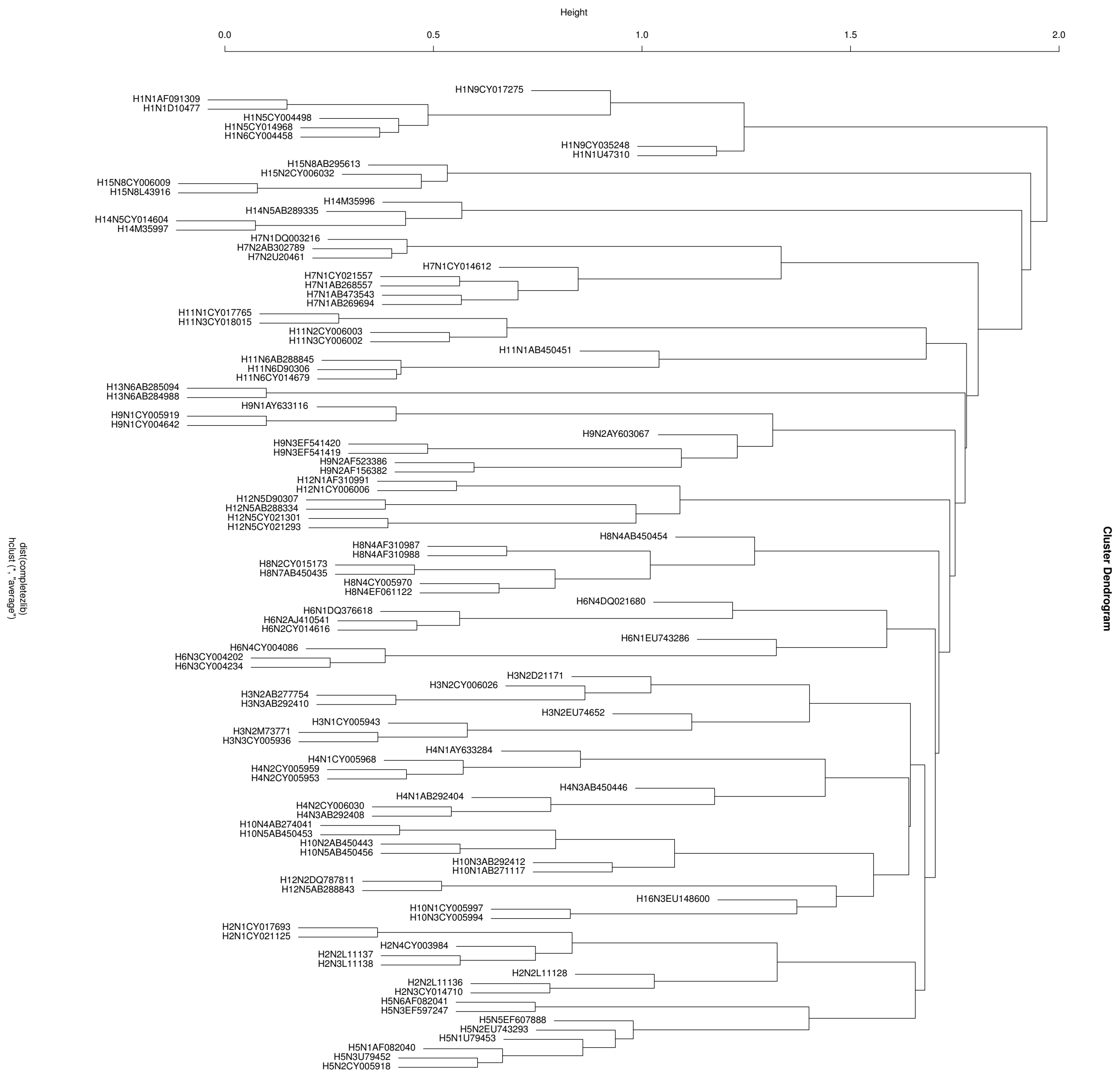

Fig. 9. Clustering of all HA sequences via `hclust`; compr.: `zlib`

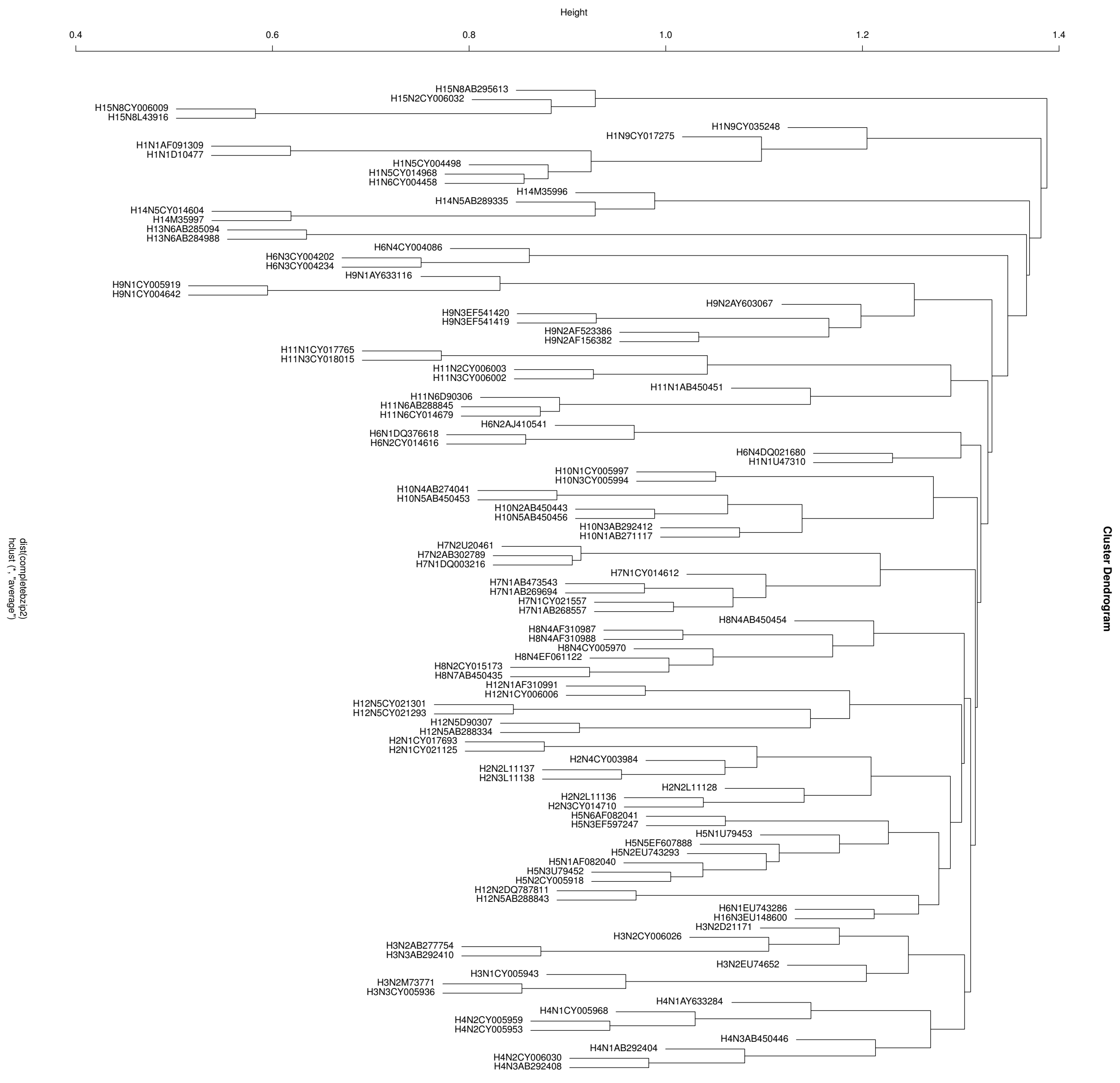

Fig. 10. Clustering of all HA sequences via `hclust`; compr.: `bzip`