sciendo

DOI: 10.2478/bile-2020-0008

Biometrical Letters
Vol. 57 (2020), No. 2, 89-118

"Old Techniques for New Times": the RMaCzek package for producing Czekanowski's diagrams

Krzysztof Bartoszek[1], **Albin Västerlund**[2]

[1]Department of Computer and Information Science, Linköping University, 581 83 Linköping, Sweden, e-mail: krzysztof.bartoszek@liu.se, krzbar@protonmail.ch
[2]Department of Computer and Information Science, Linköping University, 581 83 Linköping, Sweden, e-mail: abbe–93@hotmail.com

Summary

Inspired by the **MaCzek** Visual Basic program we pprovide an R package, **RMaCzek**, that produces Czekanowski's diagrams. Our package permits any seriation and distance method the user provides. In this paper we focus on the `"OLO"` and `"QAP_2SUM"` methods from the **seriation** package. We illustrate the possibilities of our package with three anthropological studies, one socioeconomic study and a phylogenetically motivated simulation study.

Key words: Czekanowski's diagram, distance matrix visualization, matrix reordering, seriation

1. Introduction

Clustering methods are today a standard way of summarizing even complicated data sets. Multiple algorithms and approaches are possible (for an introductory overview see e.g. Kogan, 2007), starting with the classical k-means algorithm (Steinhaus, 1956). Alternatively to visualizing the originally measured values, one may present observations using the distances between them, e.g. through multidimensional scaling plots (e.g. Ch. 4, Everitt and Hothorn, 2011). All of these are possible due to the advancements in digital electronic computers—permitting rapid calculations and high-resolution colour presentation of data. This was not always the case. At the beginning of the previous century high-dimensional data was also collected, but its presentation was restricted to usually black-and-white, manually created figures on paper. With these constraints, the Polish anthropologist Czekanowski (1909) proposed what are now called Czekanowski's

diagrams. To construct such a diagram one only needs to be able to calculate the distance between observations. Then, the next step is to find an ordering of the data points so that those with small distances are close to each other; this is a seriation problem (Hahsler et al., 2008). Finally, one plots a matrix (rows and columns correspond to observations) where each cell contains a symbol representing the distance between the appropriate pair of objects. As the number of symbols is limited, the distances have to be grouped, e.g. the interval (minimum distance, maximum distance) is split into a number of consecutive subintervals and each interval gets assigned a unique symbol (e.g. dots with varying size; see e.g. Fig. 1 or 2). Even though time-consuming, it was possible to manually present clusterings of high-dimensional observations, provided that the number of data points was not overly large. For example, Czekanowski (1909) had distances between 13 skulls belonging to various archaic humans. Each skull was characterized by 27 variables, but there were missing measurements (original data from Stołyhwa, 1908). The largest manually ordered diagram that we are aware of consists of 108 observations. Sołtysiak and Jaskulski (1999) report that such diagrams can be found in the archives of Warsaw University. These large diagrams, made carefully with a pen, are among the documents left by Bolesław Rosiński, a student of Jan Czekanowski. Andrzej Wierciński recalled (in discussions with Arkadiusz Sołtysiak) that as a young student in the 1940s and 1950s he spent a lot of time arranging and drawing these diagrams (personal communication with Arkadiusz Sołtysiak, 19 March 2019).

However, the advent of computers allowed these operations to be done automatically. In the 1980s two programs are reported to have appeared (`http://www.antropologia.uw.edu.pl/MaCzek/maczek.html`), **Ediaczek**, for botanists (has been reported in use by Cieśla, 2009) and a program of F. Szczotka (Szczotka (1972) and also Bergman (2003) reported personal communication). Then, Sołtysiak and Jaskulski (1999) developed an originally Turbo Pascal, and later Visual Basic program **MaCzek** that creates Czekanowski's diagrams and allows the user, amongst other things, to choose the distance function, seriation method (including manual rearrangement) and symbols for distance intervals (`http://www.antropologia.uw.edu.pl/MaCzek/maczek.html`). Liiv (2010) provides a review and historical perspective on these methods, and Jaskulski and Sołtysiak (2004) discuss the development and various applications, with an interesting focus on the pre-computer era, of Czekanowski's diagrams (unfortunately for most readers, it is in Polish). Even though **MaCzek** seems to be the most popular

program to create these diagrams, it seems to be confined to (predominantly Polish) anthropological (e.g. Sołtysiak, 2000), botanical (e.g. Cieśla, 2009), environmental (e.g. Boryło et al., 2012; Dołęgowska et al., 2013) and socioeconomic (e.g. Homa and Mościbrodzka, 2016; Krakowiak-Bal, 2009; Warzecha, 2015) communities. This can be attributed to the fact that **MaCzek**'s website and manual are provided only in Polish. Furthermore, **MaCzek** only runs on Windows operating systems and can handle only up to 250 observations with up to 100 variables. With contemporary big data and the multitude of software platforms, such restrictions seem prohibitive. Following encouragement from Mirosław Krzyśko and Arkadiusz Sołtysiak, Västerlund (2019) provided an R implementation of Czekanowski's diagram creation through the **RMaCzek** package (available on CRAN `https://cran.r-project.org/web/packages/RMaCzek/`). An important component of this R implementation is that any user-provided distance and seriation method can be used. By default **RMaCzek** offers the `stats::dist()` method, all (applicable) methods from the **seriation** package and a custom implemented genetic algorithm. However, the user may provide their own distance function and, using the interface offered by the **seriation** package, may use their own custom arrangement function; see Section 4.2. Furthermore, the number of observations and their dimension are only constrained by the user's memory, CPU and graphical display capabilities.

We hope that **RMaCzek**'s R implementation will popularize one of the first clustering/cluster visualization/taxonomic methods, which has at its core simplicity and compactness of presentation. In today's world it might seem at first that there is a nearly infinite capacity for displaying information and that one should rather aim at presenting more and more complexity. However, due to its inherent simplicity, one of the diagram's main advantages, over today's popular heatmaps, dendrograms and other highly sophisticated approaches, seems to be that at one glance not only the groupings can be visualized, but also the relationships between and within the groupings. An interesting example of this is given by Jaskulski and Sołtysiak (2004). They consider 1993 economic data from the "old" EU countries. Two clusters can be seen from a dendrogram (low GDP and high percentage of rural population: Greece, Portugal, Ireland and Spain; high GDP and low percentage of rural population: Belgium, the Netherlands, the UK, France, Germany, Italy, Denmark and Luxembourg). Czekanowski's diagram produces the same clusters—but furthermore shows that Spain is on the bound-

ary of these two clusters, which was completely missed by the dendrogram. Furthermore, one should not overlook the existence of situations where a low-weight graphical representation is required. Examples of these include mobile websites, low-bandwidth connections, low-resolution graphical displays, grayscale journal requirements, or sometimes even instances where an overload of colours/detail can hamper human perception. In all such cases a method designed for manual, analogue cluster creation and presentation transferred to a digital, operating-system-independent platform (as far as R permits) could be the answer. Paraphrasing the name of the conference where Sołtysiak and Jaskulski (1999) presented their **MaCzek** program—"Old Techniques for New Times".

The paper is arranged as follows. We first describe what a Czekanowski's diagram is (Section 2), then in Section 3 we discuss what is being optimized, with an example simulation study. Afterwards, in Section 4 we provide details about the **RMaCzek** R package. We end with a number of example analyses in Section 5. These are reruns of our package on data from which Czekanowski's diagrams were previously constructed, for comparison of our software with other implementations. We end with a short discussion.

2. Czekanowski's diagram

It would seem (Liiv, 2010) that Czekanowski's (1909) approach "was the first published work on one-mode data analysis that was based on the permutation of the rows and columns, complemented with color (pattern) coding for better visual perception." As already noted in the Introduction, the goal of Czekanowski (1909) was to create a matrix that best visualizes the (dis)similarities between a collection of observations. As a picture is worth a thousand words, the reader is referred to Figs. 1 or 2 to see what Czekanowski's diagram is. The entries of the matrix plotted on the diagram should correspond to the (dis)similarity between the appropriate row and column observations. However, instead of writing or colour-coding the exact distance/similarity function value, one replaces these "with simple graphic characters, e.g. white squares filled with black rectangles or black dots, of differentiated diameter" (Sołtysiak and Jaskulski, 1999). The objects most similar to each other should have the largest or most noticeable objects assigned, while the least can have nothing—a purely white square. As Sołtysiak and Jaskulski (1999) point out, by changing the range of (dis)similarity values assigned to a character, changing the number or

type of characters, one "can easily emphasize, or blur, some tendencies." The exact choice of how to divide the distance values into common symbols can be a subjective procedure (Bergman, 2003) up to the user's discretion. However, the **RMaCzek** package offers the possibility of a more objective approach. The user may declare what percentage of the distances falls into each group (by default all groups have to be equal). We feel that this is closer to Czekanowski's (1909) original proposal, which is also implemented and described towards the end of this section.

An important aspect of the diagram is its potential for simplicity. The discrete number of symbols allows easy visualization of the cluster structure at the user's preferred level of detail. This level of abstraction is controlled by the number of symbols used—the more symbols there are, the more detailed is the picture, but the fewer generalizations are directly presented.

It is important to point out that to create such a diagram one needs only to be able to calculate a distance/similarity value between a pair of observations. Sołtysiak and Jaskulski (1999) wrote that all of the variables describing an observation must be of the same type (e.g. numeric, ordinal). This was certainly true with the original distance proposed by Czekanowski (1909), the average difference (Manhattan distance divided by the number of variables), or the Euclidean distance. In fact Czekanowski's (1909) approach was most criticized for the fact that his originally proposed metric did not account for scale effects or correlations between the variables (Bergman, 2003). However, this should not be considered to be a fault of the proposed visualization method. Rather it is a "degree-of-freedom" situation where the user should appropriately pre-process the data and choose an appropriate distance function (the **RMaCzek** package allows the user to specify their own) that can, for instance, handle different variable types. For example, to take care of scale (and location) effects the data can be standardized prior to any further steps (this is a user-controlled option in **RMaCzek**), or correlations can be taken into account by using the Mahalanobis distance. Furthermore, missing values for some observations can be tolerated as long as the distance function can handle this. However, as Sołtysiak and Jaskulski (1999) warn, one must be aware that an observation with multiple missing values may be close to multiple very different objects. Hence, they suggest, based on their experience, that observations with more than 50% missing variables should be removed from further analyses.

Given the distances between the observations, the question arises how to arrange them in the rows (and in the same way in the columns). We

would desire similar observations to be close to each other in the ordering, and dissimilar ones far apart. Liiv (2010) writes that "Czekanowski did not have any formal procedure for rearrangement of the elements in the matrix; therefore, probably, visual inspection and intuition was used because the size of the dataset was also considerably small." Czekanowski (1909) does not mention how he ordered the skulls, but when one carefully inspects Czekanowski's (1909) Tables II and III, and his contemporary literature, a plausible justification can be found. Stołyhwa (1907) reports, after Schwalbe (1906), the following evolutionary timeline (of archaic humans) based on anatomical relationships (cf. Fig. 2): Neanderthal, Spy, Krapina (Schwalbe (1906) and Stołyhwa (1907) classified the last three as H. primigenius), Gibraltar, Brüx, Galley–Hill, Brünn (Schwalbe (1906) and Stołyhwa (1907) classified the last three as H. sapiens), with Pithecanthropus earlier than all of them, but uncertain if an ancestor. This ordering corresponds with minor rearrangements to Czekanowski's (1909). Furthermore, one needs to add the Kannstatt and Egisheim skulls. The final Nowosiołka skull is a focal observation whose placement was Czekanowski's (1909) goal. The skull was then hypothesized, by Stołyhwa (1908), to be related (based on an interpretation of morphological comparisons) to the Krapina–Spy–Neanderthal group. However, Czekanowski's (1909) taxonomic procedure suggests that it is rather closer related to the "H. sapiens cluster". In Section 5 we re-analyse the data using **RMaCzek** and find the same.

However, usually today's researcher will not have such in-depth knowledge, from years of scientific debate, on the observations, and contemporary data collections are significantly larger, making manual arrangement next to impossible. Furthermore, computers are not yet able to apply human intuition in such a context. Therefore, algorithmic approaches are necessary. These fall under the generic name of seriation and matrix reorder methods. Seriation is a way of reordering the sequence of observations (along a one-dimensional line) so that patterns and regularities in them are best revealed (Liiv, 2010). These patterns can be found at the between-individual level, between groups of objects, and at the global level (Liiv, 2010). Furthermore, because the result is an ordered sequence of observations along a line, one can observe the chain of associations between the objects (Liiv, 2010).

There are multiple methods of seriation; Liiv (2010) provides a historical overview of them and, for instance, the R package **seriation** (Hahsler et al., 2008) implements a number of them. The **MaCzek** program offers three seriation approaches. From its description, they are a simple one where

the two most similar objects are joined at each step, and two possible genetic algorithms. The implementation of a genetic algorithm requires the implementation of an objective function that scores how good a permutation is with respect to the distance matrix. Such an objective function should encourage similar objects to be close to each other, and distant objects far from each other—in effect producing a diagram that looks like a series of hills with peaks along the matrix's diagonal. The main clusters are arranged along the diagonal, and within each cluster we want similar diagonal arrangements. Sołtysiak and Jaskulski (1999) proposed U_m, Eq. (1), as such a function. In Section 3 we investigate this and consider an alternative approach.

It is important to underline that the seriation of the observations does not depend on the discretization of the distances. The optimal ordering is found based on the original distances, and the assignment of the symbols representing the distance classes takes place only afterwards, at the visualization stage.

Finally, all Czekanowski's diagrams that we have found in the literature are symmetric—each cell is a direct encoding (by the chosen symbols) of the corresponding entry in the distance matrix. However, this is not what Czekanowski (1909) originally suggested. Czekanowski (1909) treated each column of the distance matrix separately. Each column contains distances to a given object (let us call it the "column object"). For each column he assigned one symbol to the three observations most similar to the "column object", then the fourth most similar obtained the next symbol (recall there should be some natural ordering to the symbols in order to perceive the similarity), the fifth most similar the next symbol, and the final symbol was assigned to the sixth most similar. All the other cells in the column obtained a blank symbol. Then, similarity between the observations was assessed from the similarity of their column patterns. If two objects are similar to the same objects, then they are understood to have much in common, despite perhaps not being directly similar.

In fact Bergman (2003) writes that in his personal opinion assigning separate symbols column by column to the three most similar, then separately the fourth, fifth, sixth, and leaving the others blank is more objective than subjectively grouping distances under the same symbol. In this spirit the **RMaCzek** package allows the user to choose how to group the ranks of the similarities in each column (same grouping for each column). By default, Czekanowski's (1909) original grouping $\{\{1,2,3\},\{4\},\{5\},\{6\},\{\text{rest}\}\}$

(where $i$means the i-th most similar) is used. However, one must also remember that Czekanowski (1909) proposed his grouping based on his visualization capabilities. Therefore, today's graphical possibilities allow possibly more appropriate groupings for a given dataset.

So far we have discussed clusters in the data without actually saying how to identify them. The matrix permutation algorithms have an advantage when compared with cluster algorithms in that "no information of any kind is lost, and that the number of clusters does not have to be presumed; it is easily and naturally visible" (p. 212 Späth, 1980). While on visually inspecting Czekanowski's diagrams clusters can be clearly visible, an additional step is required to specify their boundaries. This could of course be a manual cut, but otherwise some sort of computational procedure is required. Implementing and proposing such methods is beyond the scope of this work, but an immediate approach could be to use a dendrogram construction procedure that does not re-arrange the objects. Then, cluster membership would be taken according to the appropriate number of top splits, so that the desired/observed number of clusters is obtained. In Section 3 we see how well **RMaCzek**'s different methods are able to identify the main clades for phylogenetically correlated data. A similar approach was formally proposed by Szczotka (1972) based on the so-called Wrocław taxonomic method (Florek et al., 1951). Furthermore, Havens et al. (2009) also consider clustering of ordered distance matrices. However, their visualization is symmetric, restricted to grayscale encoding of the distance matrix, and no implementation seems to be provided.

3. The objective function

As already mentioned in Section 2, in order to automatically re-order the observations one needs some sort of seriation algorithm. In their original implementation, Sołtysiak and Jaskulski (1999) scored a permutation, $\pi(\cdot)$, of the objects according to the formula

$$U_m = \frac{2}{n^2} \sum_{j=1}^{n} \sum_{i=j+1}^{n} \frac{(i-j)^2}{W_{\pi(i),\pi(j)} + 1}, \tag{1}$$

where $W_{ij} = d(a_i, a_j)$ is the distance between the objects at positions i and j. The lower the value of U_m, the better the clustering. The number of possible permutations is $n!$, hence the approach of considering all possibilities

and then choosing the one with the lowest value of U_m is only feasible for very small datasets. Therefore, a genetic algorithm optimization method, based on `GA::ga()` (Scrucca, 2013, 2017), that searches for the permutation with minimum U_m is shipped with **RMaCzek**. Furthermore, seriation methods found in the **seriation** package are also available with alternative objective functions.

Västerlund (2019) found empirically that minimizing U_m might not be the best strategy. In the course of his experiments it was observed that the `"QAP_2SUM"` (Barnard et al., 1993) consistently best minimizes U_m for a given dataset. This is not surprising as, based on the manual pages of `seriation::criterion()` and `seriation::seriate()`, `"QAP_2SUM"` minimizes, over permutations, $\pi(\cdot)$, of $(1, \ldots, n)$ the so-called 2-sum criterion

$$L_{2sum} = \sum_{j=1}^{n} \sum_{i=1}^{n} \frac{(i-j)^2}{W_{\pi(i),\pi(j)} + 1}. \tag{2}$$

Hence, as distances are symmetric, it is obvious that $L_{2sum} = n^2 U_m$ and the goals of minimization of these two objectives are equivalent to each other. However, it was also observed (subjective evaluation of resulting Czekanowski's diagrams by Västerlund, 2019) that `"QAP_2SUM"` minimization (and so U_m) tends to work by forcing objects that have a large distance between them to be far away. This might not be the best, as close-by objects might not be clustered together. On the other hand the `"OLO"` (Optimal leaf ordering, Bar-Joseph et al., 2001) method consistently did not perform best at minimizing U_m—not surprisingly as its optimization goal is something else—but produced visually more appealing diagrams. Citing the help page of `seriation::seriate()`, `"OLO"` "produces an optimal leaf ordering with respect to the minimizing the sum of the distances along the (Hamiltonian) path connecting the leaves in the given order." More formally (Earle and Hurley, 2015), the permutation $\pi(\cdot)$ of the $(1, \ldots, n)$ objects is found such that

$$L_{\text{path length}} = \sum_{i=1}^{n-1} W_{\pi(i),\pi(i+1)} \tag{3}$$

is minimized. Notice that this corresponds to finding the minimal-length Hamiltonian (each node visited exactly once) path inside the complete graph where each vertex is an observation and the branch between two vertices has length equalling the distance between the two corresponding objects.

Finding such a permutation is NP-hard, as it is the same as solving the travelling salesman problem (without returning to the origin).

From Västerlund's (2019) numerical experiments it turns out that, from our perspective, the path length minimization method's main focus is on clustering close-by objects. And it seems that in turn a better Czekanowski's diagram is produced. We compare these three methods in Tables 1, 2, 3 and provide an example visualization of a dataset illustrating that minimizing U_m might not provide the best ordering, and minimization of path length works better.

We simulate phylogenetically correlated data on a tree that has three distinct clades (clusters). The phylogenetic tree was simulated using the function `mvSLOUCH::simulate_clustered_phylogeny()`, from the package **mvSLOUCH** (Bartoszek et al., 2012). It simulates a given number of pure birth trees (using **TreeSim** Stadler, 2009, 2011) and then joins them with a pure birth tree. However, the "tip branches", i.e. those leading to the cluster, of the joining phylogeny are elongated—hence causing the clades to be distant from each other; see Fig. 1. Then, on top of the phylogeny a 10-dimensional Ornstein–Uhlenbeck (OU) process is simulated, using **mvSLOUCH**, with randomly generated coefficients; see the code in the Supplementary Material. We evaluate, over 100 repetitions, how all the three methods minimize U_m (Table 2) and the path length criterion (Table 3), how correct the clusters are (Table 1), and provide a visualization of the study and example Czekanowski's diagrams for the setup (Fig. 1). Our phylogeny has three distinct clades of size 30 tips each. Each clade has an "optimal value" of the OU process. If these optima are clade-specific, then this should make the clades more distinctive. The trait values in the clades would then exhibit similarity not only through greater evolutionary relatedness (stronger correlations), but also through mean values common to the clade. The actual values of the optima are randomly drawn from a normal distribution. We consider three scenarios. The three clades have optima drawn from Gaussians with very different means (labelled distinct optima), with different but similar means (labelled similar optima), or with equal means (labelled "equal" optima). This is seen in the different ways `mPsi` is set in the function calls below. The matrix of trait values has rows ordered according to the phylogeny—hence the `czek_matrix()` starts with the correct ordering of the observations. We also study how it behaves if the trait measurement matrix's rows are randomly shuffled. We scale the data, i.e. `scale_data=TRUE`, so the actual values of the optima do not matter,

only how different they are from each other.

To assess the correctness of the clusters, we report in Table 1 how many tips from each cluster are in the first, second and third thirty observations. As there are multiple repetitions, we make a final rearrangement of the ordering that minimizes the mismatch for the three groups (function `f_cluster_assess()`; see code in Supplementary Material). The data for Table 1 and Fig. 1 were produced by the calling `f_doRMaCzekAnalysis()` in the R code in the Supplementary Material.

The function `mvSLOUCH::simulate_clustered_phylogeny()` enhances the usual `phylo` class object with two extra fields `edges_clusters` (to which cluster does each edge belong, or to the joining part of the tree?) and `tips_clusters` (to which cluster does each tip belong?). This enhanced object then belongs to the custom `"clustered_phylo"` class. Then, such a phylogeny can be plotted directly using **mvSLOUCH**'s capabilities,

```
> phyltree<-output[[1]]$phyltree
> plot(output[[1]]$phyltree,clust_cols=c("blue","green","red"),
clust_edge.width=3,joiningphylo_edge.width=3,
show.tip.label=FALSE)
```

where the `plot()` function for a `"clustered_phylo"` class is based on `ape::plot.phylo()` (from the **ape** package, Paradis and Schliep, 2018).

We can see in Table 1 that the previously mentioned observations seemed correct. First, in the easiest case, when the optima of the three clusters are distinct, all three methods produced a completely or nearly correct clustering. When the optima became more similar, then the three methods behaved similarly, clustering on average 80% objects correctly, but it would seem that the optimal leaf ordering has a slight edge. However, optimal leaf ordering completely outperforms the other two methods in the most difficult setup—when the optimal values for the clusters are drawn from the same distribution and also the ordering of the input data was shuffled. The results in Tables 2 and 3 confirm that all three methods optimize what they are meant to optimize. The genetic algorithm and `"QAP_2SUM"` minimize U_m better than `"OLO"`, while the latter minimizes the path length criterion better than the former two. This pattern is visible in all the setups, both "easy" and "difficult". However, `"OLO"` is only slightly worse in minimizing U_m, but it completely outperforms the other methods in minimizing the Hamiltonian path length. This is again evident in both the "easy" and "difficult" setups. Table 4 illustrates another great advantage of the `"OLO"` method—it is faster than `"QAP_2SUM"`. The custom **RMaCzek** genetic al-

gorithm has an extremely long running time, and given that it does not perform any better than `"OLO"` and `"QAP_2SUM"`, it is kept only as a backup so that the package has its own internal seriation method. In Section 5 we run the `"OLO"` method on datasets that were arranged by **MaCzek**, and it would seem that while both capture the same clusters, path length minimization seems to be able to perform better arrangement within and between the clusters. This indicates that path length minimization seems to better capture the transitions between and within clusters.

Table 1. Fraction of 100 repetitions of correctly clustered observations by the `czek_matrix()` function for different simulation setups. In brackets the standard deviation of the fraction. Each cluster (clade on the phylogeny) had 30 observations (tips). The designation "Equal" does not mean that the optima were actually equal—only that they were drawn from a normal distribution with the same expected value.

Simulation type	`order="ga"`	`order="QAP_2SUM"`	`order="OLO"`
Correct input order			
Distinct optima	1, 1, 1 (0), (0), (0)	1, 1, 1 (0), (0.003), (0.003)	1, 1, 1 (0), (0), (0)
Similar optima	0.94, 0.882, 0.941 (0.092), (0.115), (0.085)	0.894, 0.826, 0.898 (0.119), (0.135), (0.115)	0.947, 0.887, 0.931 (0.09), (0.132), (0.119)
"Equal" optima	0.776, 0.657, 0.648 (0.18), (0.17), (0.152)	0.678, 0.634, 0.58 (0.16), (0.172), (0.144)	0.745, 0.707, 0.659 (0.175), (0.174), (0.181)
Shuffled input order			
Distinct optima	1, 1, 1 (0), (0.003), (0.003)	1, 1, 1 (0.011), (0.011), (0)	1, 1, 1 (0), (0), (0)
Similar optima	0.832, 0.739, 0.873 (0.136), (0.14), (0.12)	0.853, 0.765, 0.879 (0.136), (0.137), (0.124)	0.891, 0.833, 0.901 (0.133), (0.161), (0.139)
"Equal" optima	0.599, 0.567, 0.603 (0.151), (0.15), (0.161)	0.606, 0.572, 0.605 (0.152), (0.144), (0.171)	0.681, 0.659, 0.673 (0.179), (0.163), (0.187)

Table 2. Average and standard deviation of the final values of U_m associated with Czekanowski's diagrams proposed by the different methods.

Simulation type	`order="ga"`	`order="QAP_2SUM"`	`order="OLO"`
Correct input order			
Distinct optima	218.867(5.552)	219.578(13.966)	218.461(5.735)
Similar optima	221.248(4.807)	217.471(4.456)	230.426(7.438)
"Equal" optima	230.268(3.759)	223.779(2.777)	240.116(4.972)
Shuffled input order			
Distinct optima	218.956(6.002)	220.324(15.644)	219.412(5.757)
Similar optima	223.693(4.798)	218.567(3.587)	233.256(7.486)
"Equal" optima	231.526(3.803)	224.539(2.8)	240.827(4.683)

Table 3. Average and standard deviation of final values of the `"Path_length"` criterion (as returned by `seriation::criterion()`) associated with Czekanowski's diagrams proposed by the different methods.

Simulation type	`order="ga"`	`order="QAP_2SUM"`	`order="OLO"`
Correct input order			
Distinct optima	57.593(22.427)	46.197(17.772)	33.735(10.353)
Similar optima	262.733(20.803)	263.106(23.578)	161.437(9.774)
"Equal" optima	305.577(19.34)	306.774(17.704)	184.665(10.194)
Shuffled input order			
Distinct optima	52.086(17.958)	43.672(14.887)	32.666(9.213)
Similar optima	278.27(21.672)	272.043(23.911)	167.347(12.091)
"Equal" optima	316.262(17.672)	311.837(17.619)	189.062(12.371)

4. The RMaCzek package

4.1. User interface

The **RMaCzek** package offers the user two main functions `czek_matrix()` and `plot.czek_matrix()`. The first one takes the user's data (in the form of a numeric matrix, data frame or matrix of distances between observations, i.e. a `"dist"` object in the last case) and returns a matrix of dis-

Table 4. Average and standard deviation of running times in seconds for the simulation results are presented in Table 1. The simulations were run using **RMaCzek 1.2.0** in R 3.6.1 on a 3.50GHz Intel® Xeon® CPU. While absolute times might not be comparable, as other tasks were also running on the machine, the relative times immediately show that the genetic algorithm's running time is prohibitively large in comparison to the two other methods from the **seriation** package.

Simulation type	`order="ga"`	`order="QAP_2SUM"`	`order="OLO"`
Correct input order			
Distinct optima	506.275(5.529)	0.0613(0.0214)	0.041(0.013)
Similar optima	498.705(19.066)	0.057(0.015)	0.039(0.011)
"Equal" optima	440.94(4.096)	0.052(0.01)	0.036(0.006)
Shuffled input order			
Distinct optima	498.969(7.916)	0.056(0.016)	0.04(0.013)
Similar optima	506.927(4.427)	0.06(0.03)	0.046(0.026)
"Equal" optima	443.235(13.664)	0.055(0.012)	0.037(0.006)

tances between the observations (of class `"czek_matrix"`). The user is able to specify, amongst others, the seriation method to be used (parameter `"order"`), the number of classes (parameter `"n_classes"`), the intervals (parameter `"interval_breaks"`), and the distance function to be used (parameter `"distfun"`, default `stats::dist()`; of course for this parameter to make sense the input cannot be a distance matrix—if it is, then `"distfun"` is ignored). If the `"order"` parameter is `NA`, then no seriation is done and the function prepares the diagram with the user-provided ordering (equivalent to `"order"="Identity"`). Alternatively the user may provide a numeric vector (a permutation of `1:number of observations`) according to which the observations will be reordered.

Through the parameter `"scale_data"`, by default `TRUE`, the user has the option to first scale and centre the data, via `base::scale()`. The data is next transformed into a distance matrix, using a possibly user-provided distance function. Then, an ordering of the rows (and columns) of the distance matrix is found that optimizes the objective function (under the provided seriation method). The returned `"czek_matrix"` object resembles the matrix with distances between the observations. However each entry of this

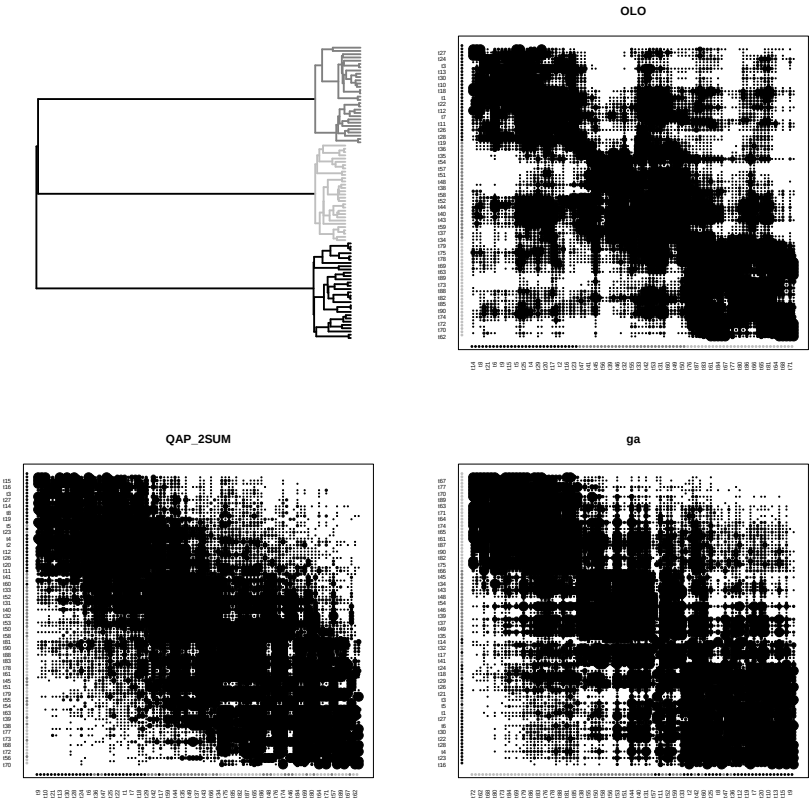

Figure 1. Example clustered phylogeny and Czekanowski's diagrams from a 10-dimensional OU process simulated on the tree. The optima values are the same as in the similar means setup. The input data is not shuffled, i.e. the values are passed in the correct ordering, prior to the seriation. The colours of the dots along the vertical and horizontal axes of the diagram correspond to the cluster to which the given observation belongs. Top right: diagram produced by `order="OLO"`; bottom left: by `order="QAP_2SUM"`; bottom right: by `order="ga"`. We can see that `order="OLO"` recovers the clusters correctly, unlike the two other methods. It is to be noted that this figure is for illustrative purposes and a number of reruns were required to obtain the above graphs; usually `"OLO"` would also have some errors, and sometimes `"QAP_2SUM"` and `"ga"` would do better than seen above. However, `"OLO"` was never (visually) observed to do worse than `"QAP_2SUM"` or `"ga"`.

object is not the distance between the observations, but an integer indicating to which distance interval the distance belongs. The lower the integer, the closer together were the observations. Importantly, in the output `"czek_matrix"` object the original order of the rows and columns is retained. The attribute `"order"` contains the optimal ordering as found by the method. It is important to point out that the vector in `"order"` is the ordering of the rows in the original data matrix (or rows/columns if the input is a `dist` object). This is particularly important to remember if the input data has row names that correspond to numbers and the original ordering is not "`1:n`".

If the user wishes, a manual rearrangement – tweaking of the observations – is possible using the provided `manual_reorder` method. This function also updates the attributes of the `"czek_matrix"` object. Alternatively, one could make the changes directly in the attribute "by hand". One then also has to remember to manually update the criteria attributes, where `"criterion_value"` is the value of the criterion under which the ordering was optimized, by default the Hamiltonian path length. Below we give an example of both approaches, switching the first two observations.

```
>library(RMaCzek)
>x<-mtcars
>czkm<-czek_matrix(x)
>neworder<-attr(czkm,"order")
>neworder[1:2]<-neworder[2:1]
>czkm_neworder<-manual_reorder(czkm,neworder,orig_data=x)
>attr(czkm,"order")<-neworder
>attr(czkm,"Um")<-Um_factor(distMatrix=dist(scale(mtcars)),
order=attr(czkm,"order"),inverse_um=FALSE)
>attr(czkm,"Path_length")<-attr(czek_matrix(x,
,order=attr(czkm,"order")),"Path_length")
>attr(czkm,"criterion_value")<-attr(czek_matrix(x,
order=attr(czkm,"order")),"Path_length")
```

Apart from a good seriation of the objects, a key step is to correctly divide the distances (or ranks of distances). By default the distances are divided so that each class has an equal number of distances (e.g. if `"n_classes"=5`, then each class will have 20% of the distances). However, the user can easily change this through the `"interval_breaks"` parameter. They may provide a vector summing to one, which will be understood as the fraction of distances in each class. Otherwise if a vector of positive numbers is provided,

then these will be the interval breaks of the distances (the vector has to start with 0 and end with the largest distance). It is important to note, after Sołtysiak and Jaskulski (1999), that different choices lead to the emphasizing of different aspects of the similarities, and one has to consider the choices carefully. As the most useful visualization problem is a data-specific perception issue and not an algorithmic one, it is advisable to try out different possibilities and choose the one(s) that best seem to capture the information in the sample. It might happen that possibilities of manual adjustment will be crucial.

The function has a number of more advanced options. As already mentioned in Sec. 2 one can choose to have an asymmetrical diagram—as originally proposed by Czekanowski (1909). This is achieved by setting the parameter `"original_diagram"=TRUE` (by default `FALSE`). Then, one can set how to group the ranks of the distances into symbols, using the parameter `"column_order_stat_grouping"`. This grouping is applied to each column. The default setting is (as in Czekanowski, 1909) to group first the three observations most similar to the column, then the fourth most similar, the fifth most similar, the sixth most similar and finally all the remaining ones (assigning the blank symbol). Here one passes a vector with the border ranks, e.g. for Czekanowski's (1909) setting this would be `c(3,4,5,6)`. In this case the `"n_classes"` parameter is ignored. The user may also mark a number of objects through the parameter `"focal_obj"`, so that they will not take part in the seriation and will be ordered last. This could be useful if there are observations that one would want to manually experiment with.

The `plot` method for the `"czek_matrix"` class plots the `"czek_matrix"` object. The user is able to control the colours, symbols and sizes of the graphics representing the distances. Furthermore, the user is given the ability to manipulate the cell sizes, the axes and labels. In Czekanowski's original proposal the distances could only be represented by different black and white symbols. In our package, for historical compatibility, by default we provide the distances as black and white symbols (but the user is also able to create a grayscale figure or a coloured figure, like a heatmap). The entries in the `"czek_matrix"` object correspond to the symbol/size of symbol/grey level/colour that is plotted between the two observations.

The `"czek_matrix"` class also has an associated `print` method. If called, it will print out the ordered objects along with the various factors associated with the ordering. If the parameter `print_raw` is set to `TRUE`, then the actual `"czek_matrix"` class object is displayed with its attributes.

4.2. Providing a custom seriation method

Apart from the custom implemented genetic algorithm, **RMaCzek** relies on the **seriation** package to provide methods to order the observations. As the package allows user-defined seriation methods, the same mechanism can be used to pass a user-defined seriation method to the `czek_matrix()` function. Below we show how the seriation method will just be a wrapper around **RMaCzek**'s inbuilt genetic algorithm. As a reference for this, the user may look at the code of the `RMaCzek:::.register_seriate_ga()` function. The input and output of the seriation function must conform to the requirements of the **seriation** interface. In particular, based on `?seriation::set_seriation_method`, the user-defined method has to have two formal arguments: `"x"`, the object for distance between observations; and `"control"`, a list with "additional information passed on from the function `seriation::seriate()`". The output has to be a list, with elements in the list corresponding to the dimensions of `"x"`. Each element has to be an object that can be coerced into a `ser_permutation_vector` object, e.g. integer vectors providing the row and column ordering.

```
>library(seriation)
>library(RMaCzek)
>my_ser_ga<-function(x,control)RMaCzek:::.seriate_ga(x,control)
>seriation::set_seriation_method(kind="dist",
name="my_seriate_ga_name",definition=my_ser_ga,
description="custom seriation method")
>x<-mtcars
>czek_matrix(x,order="my_ser_ga")
```

Notice that we provide `czek_matrix()` with the name of the seriation method and not the defining function object.

5. Example analyses

We begin illustrating the usage of the **RMaCzek** package by reconsidering Czekanowski's (1909) original skull data distance matrix. This distance matrix is included with **RMaCzek**, as `skulls_distances`. However, it should be noted that there is an error in Czekanowski's (1909) work. The distance from the Neanderthal skull to the Galley Hill skull (10.54) does not equal the distance from the Galley Hill to the Neanderthal skull (10.504). Obviously this is a minor typographic mistake, and going back to (Stołyhwa, 1908) we believe that 10.504 is probably correct. However, in the package

we wish to include the original data, hence one should symmetrize the matrix prior to using it.

```
>sym_skulls_distances[5,9]<-10.504
```

Running the code in the script `JCze1909.R` found in the Supplementary Material we obtain Fig. 2. While the ordering is very different, we can see that the same groups are retained. The Pithecanthropus and Kannstatt skulls are distinct (as Czekanowski, 1909, noticed). Then, we have the (Spy, Krapina, Neanderthal, Gibraltar) and (Brüx, Galley Hill, Nowosiołka, Brünn, Egisheim) clusters (the same as Czekanowski, 1909, discussed). We note that here (unlike in Czekanowski's (1909) presentation) Pithecanthropus is before the "Neanderthal" cluster—consistent with the timeline ordering proposed by Schwalbe (1906). The Nowosiółka skull is firmly placed within the "H. sapiens" cluster—just as Czekanowski (1909) wrote. Hence, while the same general groups are found as Czekanowski (1909) presented them, within each the observations are reordered.

We next look at Sołtysiak and Jaskulski's (1999) urns dataset (included as `urns` in **RMaCzek**), which was used to illustrate the original **MaCzek** program. Sołtysiak and Jaskulski (1999) considered nine measurements—height (WYS), rim diameter (SW), maximal diameter (MWB), bottom diameter (SD), average wall thickness (GS), average bottom thickness (GD) and three indices describing proportions of the vessel (W–A, W–B, W–D)—of fourteen urns from cremation graves excavated at Paprotki Kolonia 12 in Poland. One urn, "gr. 52–1", was removed from further analyses as it contained missing values for seven out of nine variables.

After running the `ASolPJas1999.R` script from the Supplementary Material, we obtain the ordering presented in Fig. 3. We can see that the ordering found by `order="OLO"` captures the three main clusters (and the same order between them) of Sołtysiak and Jaskulski's (1999) discovered ordering: {{gr. 2–1, gr. 5B–1, gr. 8–1, gr. 27–1, gr. 3–1, gr. 6A–4}, {gr. 46–2, gr. 66–2, gr. 64–2, gr. 72–2, gr. 23–?}, {gr. 58–3, gr. 113–1, gr. 102–2}} while "QAP_2SUM" finds only the two main clusters {{gr. 2–1, gr. 5B–1, gr. 8–1, gr. 27–1, gr. 3–1, gr. 6A–4}, {gr. 46–2, gr. 66–2, gr. 64–2, gr. 72–2, gr. 23–?, gr. 58–3, gr. 113–1, gr. 102–2}}. Despite finding the same main structure, it seems that the `"OLO"` method performs better arrangement within the clusters. This is actually evident from the "asymmetric diagrams", where the most similar are marked in each column. That for `"OLO"` is much more "diagonal" than Sołtysiak and Jaskulski's (1999) original or that for `"QAP_2SUM"`.

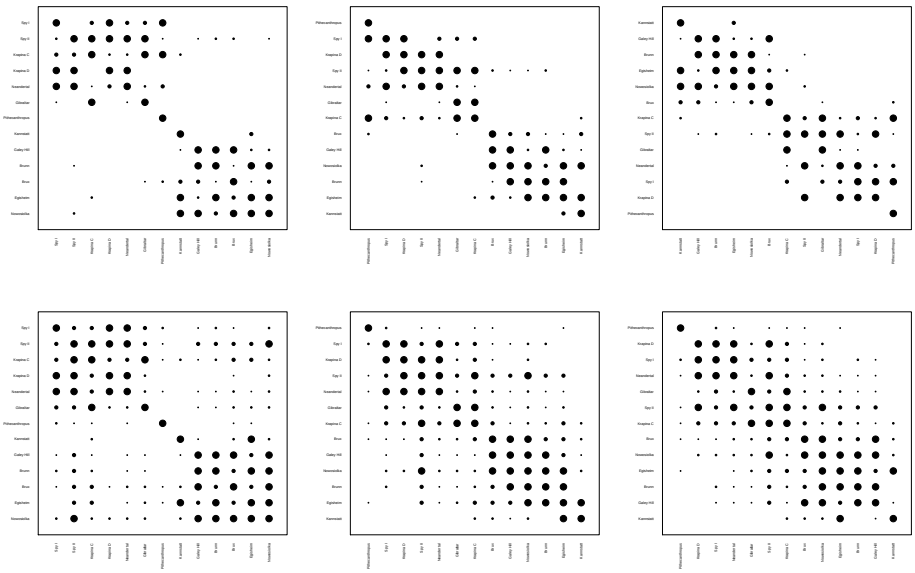

Figure 2. Visualization of Czekanowski's (1909) original distance matrix between archaic humans' skulls. Top row: diagrams with Czekanowski's (1909) original proposal presenting in each column the most similar rows; bottom row: symmetric diagram directly visualizing the distance matrix. Left column: data presented in Czekanowski's (1909) original ordering ($U_m = 3.133$, path length = 85.114); centre column: diagram produced by `order="OLO"` ($U_m = 2.56$, path length = 59.953); right column: diagram produced by `order="QAP_2SUM"` ($U_m = 2.483$, path length = 75.928).

As a third example we consider Sołtysiak's (2000) dataset of Akkadian cylinder seals depicting the Serpent God (included in the **RMaCzek** package as `seals_similarities`). This is a similarity matrix between 37 seals with 100 as maximum similarity. The original dataset consisted of 39 seals, but two (36 and 39) had to be removed "due to a lack of data in the literature" (Sołtysiak, 2000). Each seal is described by a binary vector of 22 variables; each variable is the presence or absence (or missing data due to damage to the seal) of some attribute. Examples of the attributes are "Serpent God is on the right", "end of the Serpent God's tail is clear and raised", "the Serpent God holds a goblet". Before passing the data to the `RMaCzek::czek_matrix()` function one has to set 100 on the diagonal (`NA` in the original data) and change it to a distance matrix as `100-similarity`,

```
>diag(seals_similarities)<-100
>seals_distances<-as.dist(100-seals_similarities)
```

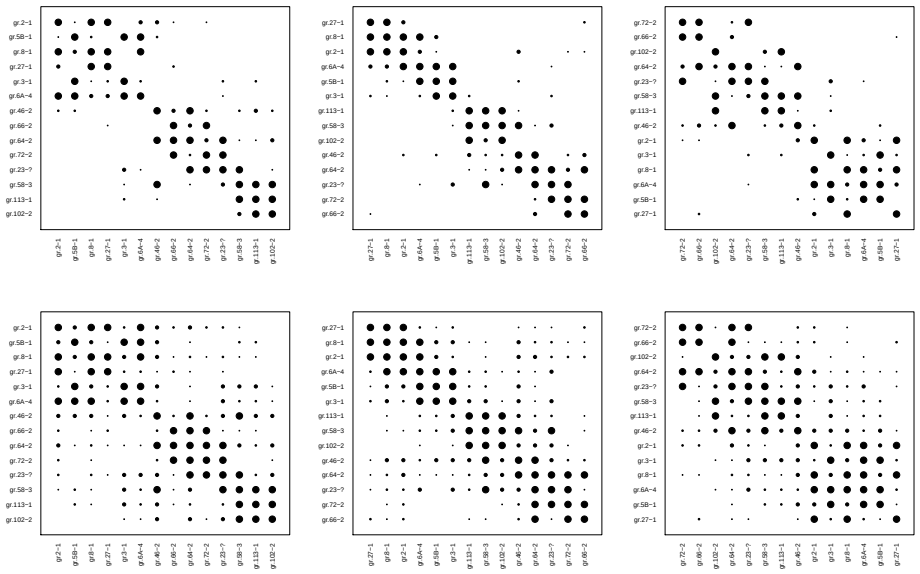

Figure 3. Visualization of Sołtysiak and Jaskulski's (1999) urns data. Top row: diagrams with Czekanowski's (1909) original proposal presenting in each column the most similar rows; bottom row: symmetric diagram directly visualizing the distance matrix. Left column: data presented by Sołtysiak and Jaskulski (1999) found by **MaCzek** ordering ($U_m = 5.933$, path length = 36.111, U_m was minimized); centre column: diagram produced by `order="OLO"` ($U_m = 5.873$, path length = 28.515, path length was minimized); right column: diagram produced by `order="QAP_2SUM"` ($U_m = 5.69$, path length = 38.772, U_m was minimized).

In order to make the symmetric graph as similar as possible to Sołtysiak's (2000) we need to set the number of classes and interval breaks appropriately. We found `n_classes=4,interval_breaks=c(0,40,60,80,100).` to be the best.

After running the script `ASol2000.R` from the Supplementary Material we can see in Fig. 4 Sołtysiak's (2000) original arrangement from the **MaCzek** program and **RMaCzek**'s `"OLO"` and `"QAP_2SUM"` arrangements. All are strongly diagonal, consistent with Sołtysiak's (2000) findings. Also it seems that the `"OLO"` method captured the main clusters that Sołtysiak (2000) did, e.g. $\{\{11, 12, 10, 13, 15, 02, 01\}, \{24, 29, 23\}\}$. Also, `"OLO"` groups together $\{16, 22, 25, 32, 38\}$, similarly as Sołtysiak (2000) did. Interestingly, in the seals dataset there are two arrangements that have the same path length. They differ by the relative arrangement of the observations $\{5, 27\}$,

$\{16, 25, 32\}$ and $\{33, 34\}$. By the U_m factor it would seem that $\{5, 27\}$, $\{32, 25, 16\}$ and $\{34, 33\}$ (symmetric diagram) is the better rearrangement, instead of $\{27, 5\}$, $\{16, 32, 25\}$ and $\{33, 34\}$ (asymmetric diagram). The `"QAP_2SUM"` method did seem to provide visually slightly less appealing diagrams—especially the symmetric one. In this case, the best similarities seem to be more scattered around the diagonal than in the `"OLO"` and Sołtysiak's (2000) original orderings. However, one also captures the clusters $\{02, 13, 19, 11, 15, 01, 12\}$ and $\{29, 24\}$.

Finally, we turn to re-analysing a dataset from a different field – socioeconomics. It is a study of Internet availability for pupils at schools in 36 counties of the Silesia Voivodeship, Poland (Warzecha, 2015). Each county

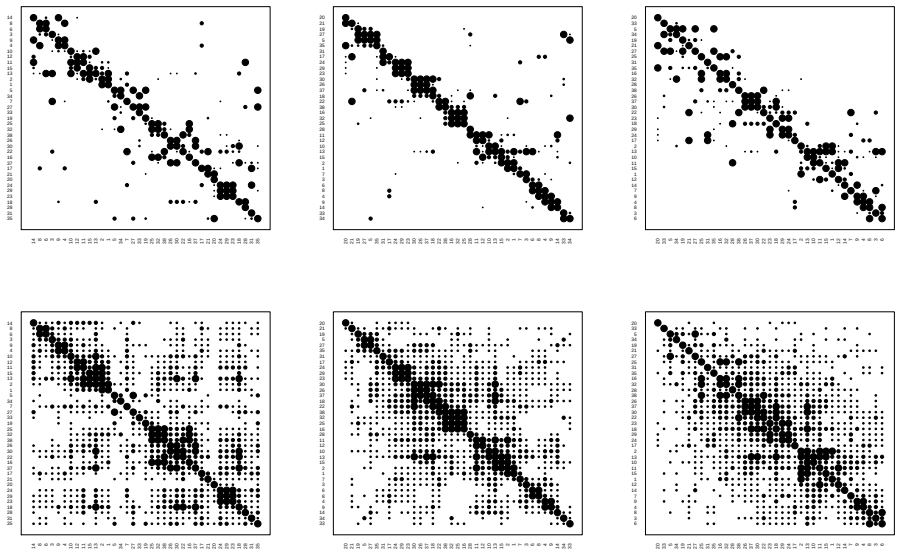

Figure 4. Visualization of Sołtysiak's (2000) seals data. Top row: diagrams with Czekanowski's (1909) original proposal presenting in each column the most similar rows; bottom row: symmetric diagram directly visualizing the distance matrix. Left column: data presented by Sołtysiak (2000) found by **MaCzek** ordering ($U_m = 3.029$, path length = 1892, U_m was minimized); centre column: diagram produced by `order="OLO"` ($U_m = 2.867$ (top); $U_m = 2.862$ (bottom), path length = 1490, path length was minimized, two orderings with identical path length were found); right column: diagram produced by `order="QAP_2SUM"` ($U_m = 2.779$, path length = 2035, U_m was minimized).

is characterized by five variables—number of students per computer with Internet access in upper secondary schools (liceum), number of students per computer with Internet access in secondary schools (gimnazjum), number of students per computer with Internet access in primary schools, proportion of upper secondary schools with computers with Internet access available to students, and proportion of secondary schools with computers with Internet access available to students. Warzecha (2015) normalized (mean centred, scaled by standard deviation) the measurements and then calculated the Euclidean distance between each pair of counties. The distance matrix between the counties is included in **RMaCzek** as `internet_availability`. In Fig. 5 we can see her ordering of the data, alongside **RMaCzek**'s ordering according to the script `KWar2015.R` in the Supplementary Material.

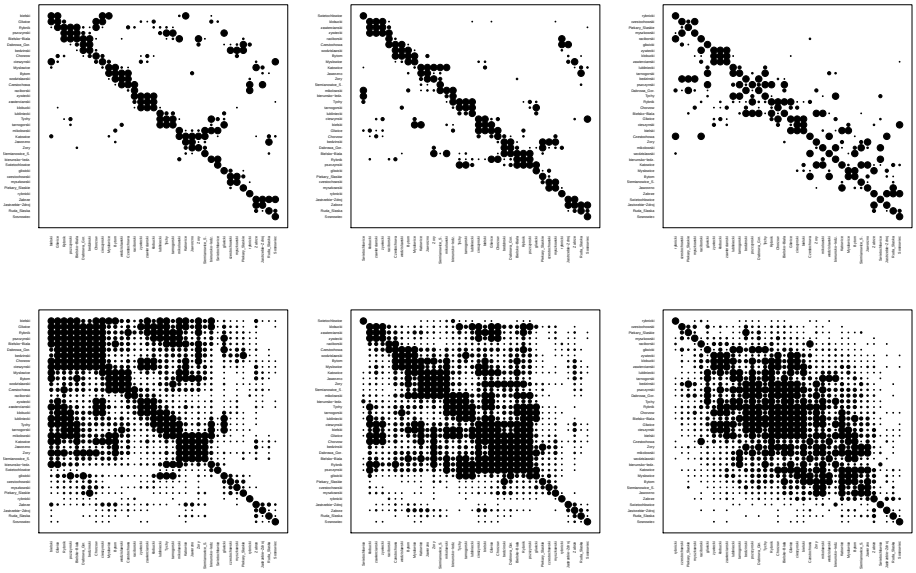

Figure 5. Visualization of Warzecha's (2015) Internet availability data. Top row: diagrams with Czekanowski's (1909) original proposal presenting in each column the most similar rows; bottom row: symmetric diagram directly visualizing the distance matrix. Left column: data presented by Warzecha (2015) found by **MaCzek** ordering ($U_m = 54.224$, path length $= 64.021$, U_m was minimized); centre column: diagram produced by `order="OLO"` ($U_m = 51.074$, path length $= 59.63$, path length was minimized); right column: diagram produced by `order="QAP_2SUM"` ($U_m = 46.769$, path length $= 81.037$, U_m was minimized).

The ordering on the labels in Fig. 5 is of course difficult to read off the axis labels, so we can display it directly

```
>print(czkm_internetavailability_OLO)
>print(czkm_internetavailability_qap2sum)
```

where the objects passed to `print()`, `czkm_internetavailability_OLO` and `czkm_internetavailability_qap2sum`, are the output of the orderings based respectively on the `"OLO"` and `"QAP_2SUM"` methods. Hence, Warzecha's (2015) original, **MaCzek**-based, ordering of the counties is (with clusters identified by her)

- cluster I: bielski, Gliwice, Rybnik, pszczyński, Bielsko–Biała, Dąbrowa Górnicza, będziński, Chorzów, cieszyński,
- cluster II: Mysłowice, Bytom, wodzisławski, Częstochowa,
- raciborski (singleton),
- cluster III: żywiecki, zawierciański, kłobucki,
- cluster IV: lubliniecki, Tychy, tarnogórski,
- cluster V: mikołowski, Katowice, Jaworzno, Żory, Siemianowice Śląskie,
- unclustered singletons: bieruńsko–lędziński, Świętochłowice, gliwicki, częstochowski, myszkowski, Piekary Śląskie, rybnicki, Zabrze, Jastrzębie–Zdrój, Ruda Śląska, Sosnowiec,

the ordering under the `"OLO"` method is

- Świętochłowice (singleton),
- cluster III: kłobucki, zawierciański, żywiecki,
- raciborski (singleton),
- cluster II: Częstochowa, wodzisławski, Bytom, Mysłowice,
- cluster V: Katowice, Jaworzno, Żory, Siemianowice Śląskie, mikołowski,
- bieruńsko–lędziński (singleton),
- cluster IV: Tychy, tarnogórski, lubliniecki,
- cluster I: cieszyński, bielski, Gliwice, Chorzów, będziński, Dąbrowa Górnicza, Bielsko–Biała, Rybnik, pszczyński,
- unclustered singletons: gliwicki, Piekary Śląskie, częstochowski, myszkowski, rybnicki, Jastrzębie–Zdrój, Zabrze, Ruda Śląska, Sosnowiec

and under the `"QAP_2SUM"` method

- unclustered singletons: rybnicki, częstochowski, Piekary Śląskie, myszkowski, raciborski, gliwicki,

- cluster III: żywiecki, kłobucki, zawierciański,
- cluster IV (part): lubliniecki, tarnogórski,
- cluster I (part): będziński, pszczyński, Dąbrowa Górnicza,
- cluster IV (part): Tychy,
- cluster I (part): Rybnik, Chorzów, Bielsko–Biała, Gliwice, cieszyński, bielski,
- cluster II (part): Częstochowa,
- cluster V (part): Żory, mikołowski,
- cluster II (part): wodzisławski,
- bieruńsko–lędziński (singleton),
- cluster V (part): Katowice,
- cluster II (part): Mysłowice, Bytom,
- cluster V (part): Siemianowice Śląskie, Jaworzno,
- unclustered singletons: Zabrze, Świętochłowice, Jastrzębie–Zdrój, Ruda Śląska, Sosnowiec.

We can see that the minimization of the Hamiltonian path length resulted in the exact same clusters that **MaCzek** identified. The singleton counties were differently placed and ordered (however, for example the county 'raciborski' is between clusters II and III). The ordering between the clusters and within the clusters is also different, but these differences would seem rather minor. In fact if one looks at the diagram in Fig. 5 it would seem that the `"OLO"` method tries to additionally build a cluster from clusters II and V. Such a cluster could have some support as it would be predominantly made up of large cities with county status. On the other hand, the `"QAP_2SUM"` method, even though it has a lower U_m factor than both **MaCzek**'s output and the `"OLO"` method, breaks up all the clusters except cluster III. The graph seems to suggest that it attempts to construct one large cluster, and drawing any conclusions from it seems difficult. In this case the asymmetric diagrams do not seem useful. Even when the values of the `column_order_stat_grouping` parameter were varied, increasing the amount of plotted points to e.g. `column_order_stat_grouping<-c(8,10,12,16)`, interpretable clusters did not start to appear. From the definition of the asymmetric method it seems that it has difficulties in usefully presenting data that has multiple clusters of varying sizes (unlike in e.g. Fig. 4, where all the clusters were of similar size).

6. Discussion

In all of the analyses run it is important to point out that the user might obtain a different ordering when rerunning the code. This was especially noticed when minimizing U_m with the `"QAP_2SUM"` method. Two runs (for original and symmetric diagrams) often resulted in slightly different orderings; the one with better U_m was chosen. All of the studies were repeated a number of times and the rerun with the best values of the criteria was retained (with a save of the random seed, provided in the Supplementary Material). It can also happen that more than one ordering has the same path length, especially when there are more observations. In such cases perhaps another criterion, e.g. U_m, should be used to differentiate between the best orderings. This occurred in the analysis of Sołtysiak's (2000) seals data. There were two arrangements that differed in the relative ordering of observations (but these were jointly in the same place in both orderings found); Fig. 4. However, the ordering visualized by the symmetric diagram had a slightly better value of U_m than the one presented in the asymmetric diagram.

In general Czekanowski's (1909) original asymmetric diagram seems to provide a better visualization of the similarity of the ordered observations in the example analyses where all the clusters are of similar size. However, this was because there were only six non-blank symbols per column. Hence, a significant amount of information is lost—and we are not distracted by spurious similarities, if the main ones are focused around the diagonal. On the other hand, when considering Sołtysiak and Jaskulski's (1999) urns data or Sołtysiak's (2000) seals data, it would seem that the clusters visualized by the asymmetric diagram (with the default setting for the symbols) are too small, and the symmetric diagram better captures their sizes. This issue is even more evident with Warzecha's (2015) Internet availability data. The asymmetric diagram fails to capture the grouping in an evident manner, even when the `column_order_stat_grouping` parameter was manipulated (so that it had a chance of corresponding to what the symmetric diagram was presenting). However, the reason for this could be that there were many clusters of varying sizes, while the asymmetric method takes the same number of most similar elements for each column (ignoring the magnitude of the similarities).

Furthermore, the `"OLO"` method, minimizing Hamiltonian path length, seemed to outperform U_m minimization. This was particularly evident in

the simulation study, in Section 3, when the seriation method managed to group objects into their correct clusters more successfully than the other two, especially with difficult setups. The re-analysis of Warzecha's (2015) Internet availability data makes this conclusion even more evident. Using **MaCzek**, whose genetic algorithm minimizes U_m, Warzecha (2015) found an ordering that graphically divided the counties into five clusters (and left a number of singletons). The `"QAP_2SUM"` method found (as it usually does) a lower value of U_m, but broke up the five clusters and instead attempted to create one large cluster that is "huddled" as much as possible around the diagonal. Such behaviour can also be seen in Fig. 1. As already mentioned, and as Västerlund (2019) observed, minimizing U_m tends to work by forcing objects that have a large distance between each other to be far apart. But when ordering data, what we actually want is that close-by objects should be clustered together, and Hamiltonian path length minimization seems to achieve this. Interestingly, even though **MaCzek** minimized U_m, its genetic algorithm seems to settle at a local minimum whose ordering is similar to the one minimizing the path length.

Acknowledgements

We would like to thank two anonymous Reviewers for their comments, which greatly improved this work. KB is supported by the Swedish Research Council (Vetenskapsrådet) grant no. 2017–04951. We are very grateful to Ewa Łączyńska-Bartoszek for translating Czekanowski's (1909) article from German, allowing us to implement his original method, as described in Section 2 and redo his skull study (Section 5). We would like to thank Arkadiusz Sołtysiak for many helpful comments and providing the seals and urns data, Katarzyna Warzecha for providing the Internet availability data and Mirosław Krzyśko for his initial encouragement for creating an R version of **MaCzek** at the XXIV National Conference Applications of Mathematics in Biology and Medicine in Zakopane–Kościelisko, 2018. The original version of **RMaCzek** was implemented as part of AB's master thesis in Statistics and Machine Learning "Czekanowski's Diagram: Implementing and exploring Czekanowski's Diagram with different seriation methods" (2019) written at the Division for Statistics and Machine Learning, Department of Computer and Information Science, Linköping University.

Supplementary Material

The random seeds and source code required to obtain the results in the

manuscript can be found in the repository
`https://github.com/krzbar/RMaCzek_BiometricalLetters_2020`.

References

Bar-Joseph Z., Demaine E.D., Gifford D.K., Jaakkola T. (2001): Fast optimal leaf ordering for hierarchical clustering. Bioinformatics 17(1): 22–29.

Barnard S.T., Pothen A., Simon H.D. (1993): Algorithm for envelope reduction of sparse matrices. In: Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, pages 493–502, New York.

Bartoszek K., Pienaar J., Mostad P., Andersson S., Hansen T.F. (2012): A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314: 204–215.

Bergman P. (2003): Wybrane metody odległości wielocechowych–rys historyczny (Some multivariate distnace methods—a historical perspective, in Polish). In: Metody statystyczne w antropologii, VI Warsztaty Antropologiczne im. Profesora Janusza Charzewskiego (Statistical Methods in Anthropology, the Sixth Prof. Janusz Charzewski Anthropological Workshop), Warsaw.

Boryło A., Skwarzec B., Olszewski G. (2012): The radiochemical contamination (^{210}Po and ^{238}U) of zone around phosphogypsum waste heap in Wiślinka (northern Poland). Journal of Environmental Science and Health, Part A 47(5): 675–687.

Cieśla A. (2009): Effect of hydrotechnical constructions on the Oder river on the phytosociological diversity of riparian habitats in the prawików forest. Forest Research Papers 70(2): 161–174. (in Polish).

Czekanowski J. (1909): Zur Differentialdiagnose der Neandertalgruppe. Korespondentblatt der Deutschen Gesellschaft für Anthropologie, Ethnologie und Urgeschichte XL(6/7): 44–47.

Dołęgowska S., Migaszewski Z.M., Michalik A. (2013): *Hylocomium splendens* (Hedw.) B.S.G. and *Pleurozium schreberi* (Brid.) Mitt. as trace element bioindicators: Statistical comparison of bioaccumulative properties. Journal of Environmental Sciences 25(2): 340–347.

Earle D., Hurley C.B. (2015): Advances in dendrogram seriation for applications to visualization. Journal of Computational and Graphical Statistics 24(1): 1–25.

Everitt B., Hothorn T. (2011): An Introduction to Applied Multivariate Analysis with R. Springer.

Florek K., Łukasiewicz J., Perkał J., Steinhaus H., Zubrzycki S. (1951): Sur la liason et la division des points d'un ensemble fini. Colloquium Mathematicum 2: 282–285.

Hahsler M., Hornik K., Buchta C. (2008): Getting things in order: An introduction to the R package **seriation**. Journal of Statistical Software 25(3): 1–34.

Havens T.C., Bezdek J.C., Keller J.M., Popescu M. (2009): Clustering in ordered dissimilarity data. International Journal of Intelligent Systems 24:504–528.

Homa M., Mościbrodzka M. (2016): Application of diagram methods and hierarchical agglomerative procedures to assess the risk of investment funds on the Warsaw Stock Exchange. Financial Sciences 4(29): 21–34.

Jaskulski P., Sołtysiak A. (2004): Diagram Czekanowskiego: pomysł, historia, zastosowania. Prace Naukowe Akademii Ekonomicznej we Wrocławiu. Taksonomia 11(1022): 374–383. (Czekanowski's Diagram : Idea, History, Implementations, in Polish).

Kogan J. (2007): *Introduction to Clustering Large and High–Dimensional Data.* Cambridge University Press.

Krakowiak-Bal A. (2009): Multidimensional comparative analysis of other gainful activities of agricultural holdings in EU countries. Infrastruktura i Ekologia Terenów Wiejskich 7: 129–137.

Liiv I. (2010): Seriation and matrix reordering methods: An historical overview. Statistical Analysis and Data Mining 3(2): 70–91.

Paradis E., Schliep K. (2018): **ape** 5.0: an environment for modern phylogenetics and evolutionary analyses in R. Bioinformatics 35: 526–528.

Schwalbe G.A. (1906): Studien zur Vorgeschichte des Menschen. E. Nägele, Stuttgart.

Scrucca L. (2013): **GA**: A package for genetic algorithms in R. Journal of Statistical Software 53(4):1–37.

Scrucca L. (2017): On some extensions to **GA** package: hybrid optimisation, parallelisation and islands evolution. The R Journal 9(1): 187–206.

Sołtysiak A. (2000): Przedstawienie Boga–Węża na pieczęciach cylindrycznych z okresu akadyjskiego: analiza ikonograficzna. Studia i Materiały Archeologiczne 10: 189–214. (Depiction of the Serpent God on cylinder seals from the Akkadian era: an iconographic analysis, in Polish).

Sołtysiak A., Jaskulski P. (1999): Czekanowski's diagram. a method of multidimensional clustering. In: New Techniques for Old Times. CAA 98. Computer Applications and Quantitative Methods in Archaeology. Proceedings of the 26th Conference, Barcelona, March 1998, number 757 in BAR International Series, pages 175–184, Oxford.

Späth H. (1980): Cluster Analysis Algorithms for Data Reduction and Classification of Objects. Ellis Horwood, Chichester.

Stadler T. (2009): On incomplete sampling under birth-death models and connections to the sampling-based coalescent. Journal of Theoretical Biology 261(1): 58–68.

Stadler T. (2011): Simulating trees with a fixed number of extant species. Systematic Biology 60(5): 676–684.

Steinhaus H. (1956): Sur la division des corps matériels en parties. Bulletin de l'Academie Polonaise des Sciences IV(12): 801–804.

Stołyhwa K. (1907): Czy H. primigenius stanowi gatunek odrębny od H. sapiens? Światowit 8: 10–34. (Is H. primigenius a separate species from H. sapiens?, in Polish. Reprint of "Homo primigenius appartient–il a une espece distincte de Homo sapiens", published in l'Anthropolgie in 1908).

Stołyhwa K. (1908): Czaszka z Nowosiółki jako dowód istnienia w okresie historycznym kształtów pokrewnych z Homo primigenius. Rozprawy Wydziału matematyczno–przyrodniczego Akademii Umiejętności XLVIII(B): 1–27. (The skull from Nowosiółka as proof of existence during the era of history shapes common with Homo primigenius, in Polish).

Szczotka F.A. (1972): On a method of ordering and clustering of objects. Applicationes Mathematicae (Zastosowania Matematyki) XIII(1): 23–34.

Västerlund A. (2019): Czekanowski's Diagram: Implementing and exploring Czekanowski's Diagram with different seriation methods. PhD thesis, Division for Statistics and Machine Learning, Department of Computer and Information Science, Linköping University.

Warzecha K. (2015): The use of quantitative methods in research on selected behavioral addictions of young people. Studia Ekonomiczne 247: 121–139.