Development of Online Learning Material for Data Science Programming Using 3D Puzzle

Naoki Yamamoto, Akio Ishida, Kazuki Ogitsuka, Nobuhiro Oishi, and Jun Murakami

***Abstract*—In multidimensional data processing, one of the important data structures is a higher-order tensor or a multidimensional array. In general, the processing related to the higher-order tensor is so complicated that we have been developing understanding support tools for it using 3D puzzles from the viewpoint of making students interested. However, although these tools have been tried by students in graduation studies and other some occasions, their introduction into lectures was one issue. Therefore, in this study, we developed a new programming exercise material for the higher-order tensor, which is supposed to be used in data science subjects, by using a 3D puzzle. This learning material is also composed of Microsoft Teams, and students can access the material remotely to learn programming. In this paper, several students actually tried this material. As a result, it was found that the students themselves could create and submit assignment reports by viewing explanatory videos and performing exercises. From this, it is expected that this material will be able to introduce to data science courses, including online use.**

***Index Terms*—Higher-order tensor, 3D puzzle, data science education, R, programing materials, remote exercise.**

I. Introduction

The higher-order tensor (multidimensional array) is one of the typical data structures that handle multidimensional data such as big data processing. And it has been applied to data reduction and pattern recognition using tensor decomposition [1].

So far, we have developed teaching materials to help understand higher-order tensors and related data processing [2]-[4]. In these developed materials, 3-D puzzles are used as a subject to arouse the interest of students, such as mathematical 3D puzzles, Rubik's Cube, Instant Insanity, and MacMahon Cube. These puzzles are expressed by higher-order tensors and are used as those materials. These materials have been tried to use for higher-order tensor programming education for graduation research students and advanced course students, and for multi-dimensional data processing introductive education for other college students and junior high school students (on the occasion of Open Campus day).

However, the regular introduction of those materials into lectures has not been done yet. In this research, we have developed a new programming learning material using a 3D puzzle, assuming that it will be used in data science related lectures. In addition, the development was carried out considering remote usage online, so the terminology of learning materials was used in the title. Note that R language [5] is used here as the programming language.

In the following sections, at first, an outline of a 3D puzzle and its solution method, and an example of its basic solution will be shown. Next, the configuration and contents of the developed programming learning material will be described in detail. Finally, the results of using this material for several students will be described.

II. 3D Puzzles and Higher-Order Tensors

A. Puzzle Problem and Its Solution

In this research, we develop learning materials for programming exercises of higher-order tensors. A higher-order tensor means a multidimensional array in this paper. The 3D puzzle used for the learning material to be developed here is as follows.

[Puzzle Problem (Light Bulb Placement Puzzle [6])]

Suppose that one light bulb can be placed in each cell of an $m \times m \times m$ cube, and m^2 pieces of it are appropriately arranged and lighted. There are special arrangement ways that every cells of the cube seems to be lit when you viewed from either of the three axis directions of Cartesian coordinates. Find how many these special arrangements there are. However, if the faces of the cube are swapped and become the same solution, they are not treated as different solutions.

(End of problem)

The learning material uses the solution of the above problem for the case of $m = 3$ and 4 to create examples and exercise problems. For reference, the following describes the solution algorithm using a personal computer for $m = 4$ and shows the solution.

Fig. 1 shows a $4 \times 4 \times 4$ cube represented by a third-order tensor (3D array) of the same size. The elements of $\mathcal{A}$ are specified using the 1st to 3rd subscripts, and each represents the row, column, and depth direction. In the figure, serial

Manuscript received October 4, 2020; revised January 15, 2021. This work was supported by the JSPS KAKENHI (Grants-in Aid for Scientific Research) under Grant No. JP18K11596.

N. Yamamoto, K. Ogitsuka, and J. Murakami are with the Department of Human-Oriented Information Systems Engineering, Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: naoki@kumamoto-nct.ac.jp, hi16ogitsuka@g.kumamoto-nct.ac.jp, jun@kumamoto-nct.ac.jp).

A. Ishida is with Faculty of Liberal Arts, Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: ishida@kumamoto-nct.ac.jp).

N. Oishi is with the Department of Information, Communication and Electronic Engineering, Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: oishi@kumamoto-nct.ac.jp).

doi: 10.18178/ijiet.2021.11.4.1505

numbers from 1 to 64 are displayed in the upper right of each element for easy understanding. If the element of $\mathcal{A}$ is 1, there is a light bulb there, and if it is 0, there is no light bulb.

TABLE I: SOLUTION FOR A $4\times4\times4$ CUBE IN FIG. 1

Solution No.	Placement of 1
1	1,6,11,16,18,21,28,31,35,40,41,46,52,55,58,61
2	1,6,11,16,18,21,28,31,35,40,42,45,52,55,57,62
3	1,6,11,16,18,21,28,31,36,39,42,45,51,56,57,62
4	1,6,11,16,18,23,28,29,35,40,41,46,52,53,58,63
5	1,6,11,16,18,23,28,29,36,37,42,47,51,56,57,62
6	1,6,11,16,18,24,25,31,35,37,44,46,52,55,58,61
7	1,6,11,16,18,24,25,31,36,39,42,45,51,53,60,62
8	1,6,11,16,19,21,28,30,36,39,42,45,50,56,57,63
9	1,6,11,16,19,24,25,30,34,37,44,47,52,55,58,61
10	1,6,11,16,19,24,25,30,34,39,44,45,52,53,58,63
11	1,6,11,16,19,24,25,30,36,39,42,45,50,53,60,63
12	1,6,11,16,19,24,26,29,34,37,44,47,52,55,57,62
13	1,6,11,16,19,24,26,29,36,39,41,46,50,53,60,63
14	1,6,11,16,20,23,26,29,34,37,44,47,51,56,57,62
15	1,6,11,16,20,23,26,29,34,40,41,47,51,53,60,62
16	1,6,11,16,20,23,26,29,35,37,44,46,50,56,57,63
17	1,6,11,16,20,23,26,29,35,40,41,46,50,53,60,63
18	1,6,12,15,18,21,27,32,36,39,42,45,51,56,57,62
19	1,6,12,15,18,23,25,32,36,37,43,46,51,56,58,61
20	1,6,12,15,18,24,27,29,36,39,41,46,51,53,58,64
21	1,6,12,15,19,24,25,30,36,39,42,45,50,53,59,64
22	1,6,12,15,19,24,26,29,34,37,43,48,52,55,57,62
23	1,6,12,15,19,24,26,29,34,39,41,48,52,53,59,62
24	1,6,12,15,19,24,26,29,36,37,43,46,50,55,57,64
25	1,6,12,15,19,24,26,29,36,39,41,46,50,53,59,64
26	1,6,12,15,20,23,25,30,34,37,43,48,51,56,58,61
27	1,6,12,15,20,23,25,30,34,40,43,45,51,53,58,64
28	1,6,12,15,20,23,25,30,35,37,42,48,50,56,59,61
29	1,6,12,15,20,23,26,29,34,37,43,48,51,56,57,62
30	1,7,10,16,19,24,25,30,34,37,44,47,52,54,59,61
31	1,7,10,16,19,24,25,30,36,38,43,45,50,53,60,63
32	1,7,10,16,20,22,27,29,34,37,44,47,51,56,57,62
33	1,7,10,16,20,22,27,29,34,40,41,47,51,53,60,62
34	1,7,10,16,20,22,27,29,35,37,44,46,50,56,57,63
35	1,7,10,16,20,22,27,29,35,40,41,46,50,53,60,63
36	1,7,12,14,20,22,25,31,34,40,43,45,51,53,58,64
37	1,7,12,14,20,22,27,29,35,37,42,48,50,56,57,63
38	1,8,11,14,20,23,26,29,35,38,41,48,50,53,60,63
39	2,5,12,15,17,22,27,32,36,39,42,45,51,56,57,62
40	2,5,12,15,17,23,26,32,36,38,43,45,51,56,57,62
41	2,5,12,15,20,22,27,29,33,39,42,48,51,56,57,62
42	2,5,12,15,20,23,26,29,33,38,43,48,51,56,57,62

As seen from the example in Fig. 1, if an 1 is placed anywhere in the first row on any side of $\mathcal{A}$ (front side in the case of Fig. 1), another 1 cannot be placed anywhere else in this row and column (also in the depth direction). There are four ways to place 1 in this case. Next, for the second row, 1 can be placed only in the column other than the column with 1 in the previous row, so there are three ways to place it. In this way, the position where 1 can be placed decreases as the number of rows increases, so there are $4!=24$ ways to place 1 on this side. Therefore, for a 3D array composed of four 2D arrays of size 4×4, there are $(4!)^4=331776$ ways to place 1. Then, the solution candidates satisfying the condition (sixteen 1's are visible on each side viewed from three directions) are found by the whole search. The problem solution can be obtained from these candidate solutions by removing the same ones when rotated in multiples of 90 degrees around three axes.

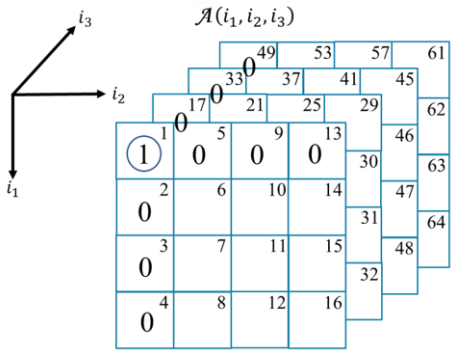

Fig. 1. Third-order tensor of size $4\times4\times4$ and its element number.

We implemented this algorithm in R and sought the solution and obtained 42 solutions shown in Table I [7]. In this table, the sequence of numbers in the right column of each solution represents the element number assigned in Fig. 1, and the light bulbs are placed at these positions and turned on.

Similarly, in the case of $m=3$ shown in Fig. 2, two solutions in Table II were obtained by calculation using R [8]. In this case, there are two solutions, and the nine light bulbs are placed in the cube at the positions indicated in the table.

In this research, the solution in the case of $m=3$ is used as an example of the programming exercise for multidimensional data processing, and that in the case of $m=4$ is used as the exercise problem.

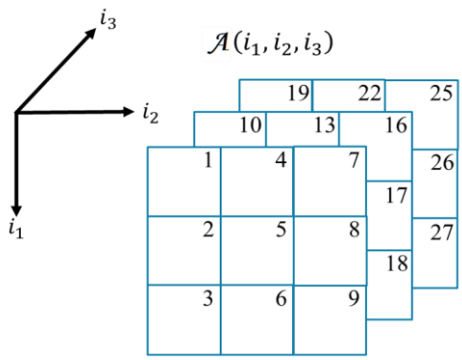

Fig. 2. Third-order tensor of size $3\times3\times3$ and its element number.

B. Definition of Higher-Order Tensor and Related Operations

1) Higher-order tensor

TABLE II: SOLUTION FOR A $3\times3\times3$ CUBE IN FIG. 2

Solution No.	Placement of 1
1	1,5,9,11,15,16,21,22,26
2	1,6,8,12,14,16,20,22,27

In this study, a higher-order tensor means a multidimensional array, as mentioned above. For example, a first-order tensor is a vector, a second-order tensor is a matrix, and a third-order tensor is a three-dimensional array. In our developed material, the 3D puzzle described in the previous subsection can be expressed by the higher-order tensor defined below [9].

[Definition 1. N-th Order Tensor]

An N-th order tensor $\mathcal{A}\in\mathbf{R}^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$ with a size of $I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N$ is defined by the following equation.

$$\mathcal{A}=\left(a_{i_1i_2\cdots i_n\cdots i_N}\right), \tag{1}$$

$$(i_1=1,2,\ldots,I_1;\cdots;i_n=1,2,\ldots,I_n;\cdots;i_N=1,2,\ldots,I_N),$$

where $a_{i_1i_2\cdots i_n\cdots i_N}$ denotes an $(i_1,i_2,\cdots,i_n,\cdots,i_N)$-th element of $\mathcal{A}$. Besides, the notation $\mathbf{R}$ expresses a set of real numbers.

(End of definition)

In (1), the subscripts $i_1,i_2,\cdots,i_N$ of $\mathcal{A}$ correspond to directions of the tensor called "mode". For instance, in the third-order tensor shown in Fig. 1, the subscript i_1 corresponds to a row of the tensor called "1-mode".

2) Sum of partial tensors

To check solutions of the 3D puzzle, in the third-order tensor of Fig. 1, it is necessary to confirm that light bulbs light up on all sides when viewed from 3 directions. To determine whether each candidate solution is a true solution or not, a sum of partial tensors defined below is used.

[Definition 2. Sum of Partial Tensors on "n-mode" of N-th Order Tensor]

A partial tensor when the n-th subscript of N-th order tensor $\mathcal{A}$ in Definition 1 is fixed to $i_n=\alpha$ is $\mathcal{A}_{i_n=\alpha}=\left(a_{i_1i_2\cdots\alpha\cdots i_N}\right)$. Now, for $i_n=1,2,\ldots,I_n$, the sum of partial tensors from $\mathcal{A}_{i_n=1}$ to $\mathcal{A}_{i_n=I_n}$ is called the sum of partial tensors on n-mode of $\mathcal{A}$, and it is defined below.

$$\mathcal{S}_{\Sigma i_n}\overset{\text{def}}{=}\mathcal{A}_{i_n=1}+\mathcal{A}_{i_n=1}+\cdots+\mathcal{A}_{i_n=I_n}. \tag{2}$$

(End of definition)

3) Matrix unfolding

In this learning material, positions of the light bulbs in the cube are expressed by a two-dimensional map. To create the map, a matrix unfolding defined below is used [9].

[Definition 3. n-mode Matrix Unfolding of N-th Order Tensor]

Operation of expanding each element $a_{i_1i_2\cdots i_n\cdots i_N}$ of N-th order tensor $\mathcal{A}$ into an (i_n,j_n)-th element of a matrix $\mathbf{A}_{(n)}$ is called the n-mode matrix unfolding of $\mathcal{A}$. The size of the matrix $\mathbf{A}_{(n)}$ is

$$I_n\times(I_{n+1}I_{n+2}\cdots I_NI_1I_2\cdots I_{n-1}),$$

where j_n is given by the following equation.

$$\begin{aligned}j_n=\;&(i_{n+1}-1)I_{n+2}I_{n+3}\cdots I_NI_1I_2\cdots I_{n-1}\\&+(i_{n+2}-1)I_{n+3}I_{n+4}\cdots I_NI_1I_2\cdots I_{n-1}+\cdots\\&+(i_N-1)I_1I_2\cdots I_{n-1}+(i_1-1)I_2I_3\cdots I_{n-1}\\&+(i_2-1)I_3I_4\cdots I_{n-1}+\cdots+i_{n-1}.\end{aligned} \tag{3}$$

(End of definition)

To put it simply, the matrix unfolding is the operation of converting a higher-order tensor into a second-order tensor (matrix). The third-order tensor used to express the 3D puzzle is transformed into three types of 1-mode matrix unfolding $\mathbf{A}_{(1)}$, 2-mode matrix unfolding $\mathbf{A}_{(2)}$, and 3-mode matrix unfolding $\mathbf{A}_{(3)}$,whose number of rows are I_1, I_2 and I_3, respectively.

4) Folding

This learning material also contains an exercise to restore the original cube form from the map of the light bulb positions created by the matrix unfolding. The operations defined below are used to do that.

[Definition 4. Folding of n-mode Matrix Unfolding into N-th Order Tensor]

Moving each element $a_{i_n j_n}$ in $\mathbf{A}_{(n)}$ of Definition 3 to an $(i_1,i_2,\cdots,i_n,\cdots,i_N)$-th element of $\mathcal{A}$ is called the folding of $\mathbf{A}_{(n)}$ into $\mathcal{A}$, where j_n is given by (3).

(End of definition)

That is, Definition 4 defines the reverse operation defined in Definition 3.

5) Tensor package of R

This learning material is for practicing programming related to higher-order tensors, and R is used as a programming language. In the exercise, each student installs and uses rTensor, which is a tensor calculation package of R [10], on their personal computer.

rTensor is provided with functions related to various tensor operations including ones shown below.

1) as.tensor: Function that creates a tensor object from an array, a matrix, and a vector.
2) modeSum: Function that computes the sum of partial tensors for a particular mode of a higher-order tensor.
3) unfold: Function to compute the matrix unfolding of a higher-order tensor.
4) fold: Function to fold the matrix unfolding into a higher-order tensor.
5) ttm: Function for calculating a product of a higher-order

tensor and a matrix.

6) hosvd: Function to calculate Higher-Order Singular Value Decomposition (HOSVD) [9] of a higher-order tensor.

In addition, the rTensor can be downloaded and used from the site called the Comprehensive R Archive Network (CRAN). In this learning material, the above-mentioned functions 1) to 4) are used to implement the operations defined in Definitions 1 to 4, respectively.

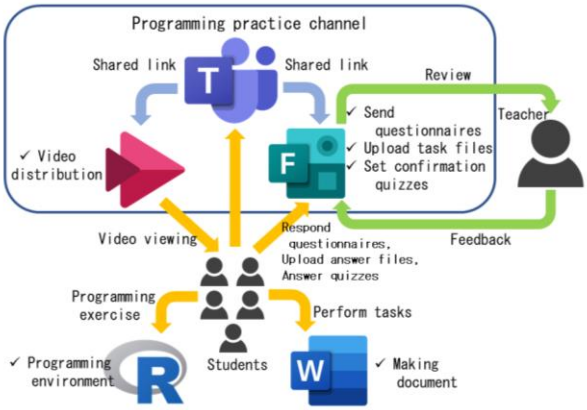

Fig. 3. Configuration of our learning material system.

III. Developed Programming Learning Material System

A. Configuration and Usage of the Learning Material System

As shown in Fig. 3, our materials use Microsoft (MS) Teams and are constructed by creating channels for programming exercises within the team. The contents of the exercises are created for each thread in the channel. In each thread, how to proceed with the exercise is explained, and the sharing link of the distribution destination of the exercise explanation video using MS Stream, the upload destination of the answer file using MS Forms, and the quiz question distribution source are created.

The learners work on the programming exercises while watching the videos explaining the exercises according to the learning procedure in each thread. They can receive teacher's review after submitting a report of the created script and execution result through MS Forms.

B. Contents of Learning Material

This learning material was developed for the subjects such as data science, assuming the learning of programming related to higher-order tensors. R is a programming language developed for statistical analysis, and we have been using it for exercises in those subjects. In this learning material as well, programming exercises using R were incorporated into the contents.

Learning using the material proceeds in the order of (a) to (f) below.

[Contents of Learning Material]

(a) Explanation of contents of R programming practice exercises and how to proceed with learning.

(b) R programming exercise I (Outline of higher-order tensors and installation of the tensor package).

(c) R programming exercise II (Creation of higher-order tensors and calculation of the sum of partial tensor).

(d) R programming exercise III (Matrix unfolding of higher-order tensor).

(e) R programming exercise IV (Folding into higher-order tensor).

(f) Confirmation quizzes.

In (a) above, a general description of the exercises is given. From (b) to (e), students do programming exercises while listening to the explanatory videos on MS Stream and create reports. MS Forms will be used to answer questionnaires and submit reports. In (f), quizzes for confirming whether the learning contents up to this point have been acquired are given through Forms, so the students submit the answer. In the next section, details about this material will be given.

C. Details of the Learning Material

This material is organized on a channel within a team of MS Teams, and each content is created per thread.

First, Fig. 4 shows a thread having the content (a) described in the previous section. This thread gives a general description of this exercise. In the figure, the contents are shown in the first half of the thread, and the points of how to proceed with the exercise are shown in the latter half.

In addition, this figure is a translation of the actual PC screen display in English, and the contents are the same as those described in the text (the same hereafter).

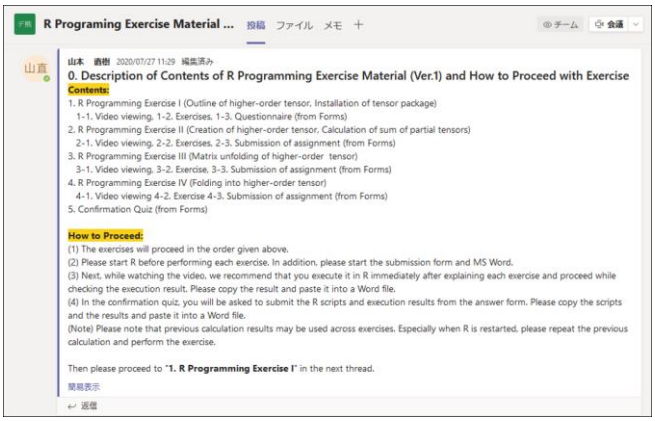

Fig. 4. Thread to explain the whole exercises.

Fig. 5 shows a thread for Programming Exercise I mentioned in the previous section. In this thread, a high-order tensor is outlined, and an exercise of installing rTensor, which is a R package for the tensor, is performed. In the first half of the thread of the figure, how to proceed with the exercise is shown, and advice is given to start R and a questionnaire form before the exercise and proceed with the exercise while watching a video. In the latter half of the thread, there are shared links to the distribution destination of explanation videos and the response destination of a questionnaire, and students can access these link destinations to proceed with the exercise.

Fig. 6 shows a screen shot of the distributed explanation video and explain an exercise regarding installing rTensor. Immediately after viewing the explanation video, the students can proceed with the learning by performing the exercises in the R environment on their PCs.

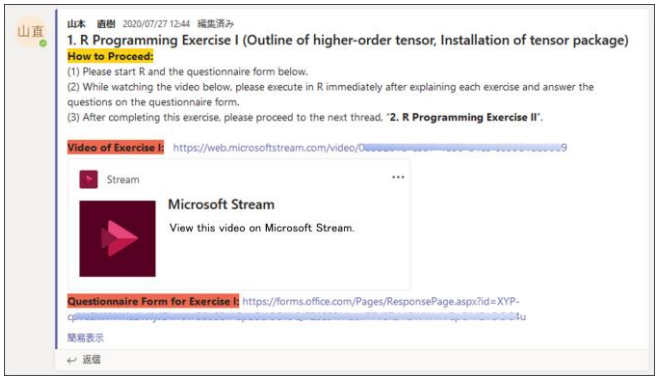

Fig. 5. Thread of Programming Exercise I.

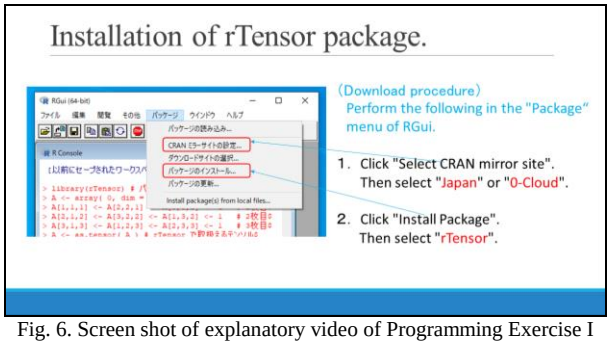

Fig. 6. Screen shot of explanatory video of Programming Exercise I

Fig. 7 shows a questionnaire form for this exercise created with MS Forms. This form asks students if they could understand an overview of the higher-order tensor and if rTensor could be installed. The details of the questions in this form are described in the next section.

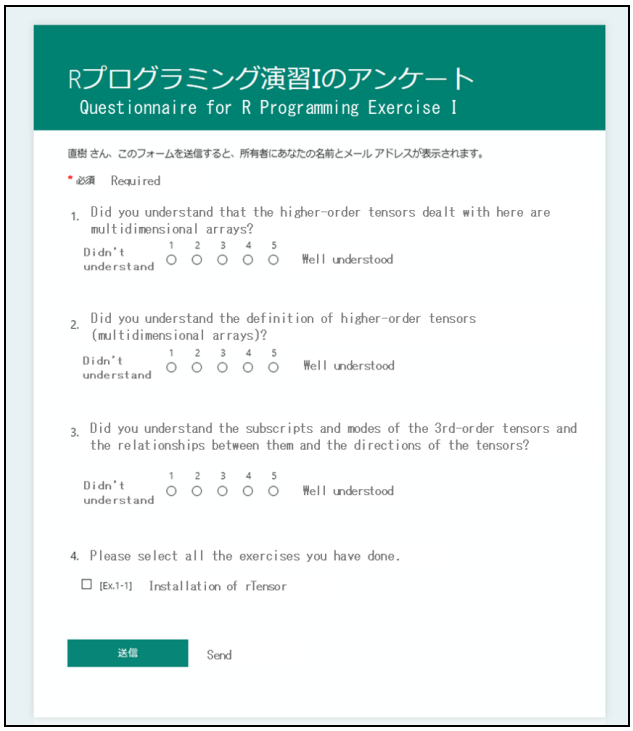

Fig. 7. Questionnaire form of Programming Exercise I.

Fig. 8 shows a thread of Programming Exercise II in the previous section. In this exercise, programming for generation of a higher-order tensor and calculation of the sum of partial tensors is performed. The structure of the thread is the same as in the Exercise I. There is an explanation of how to proceed with the exercise, and sharing links are provided for an explanation video and a submission form.

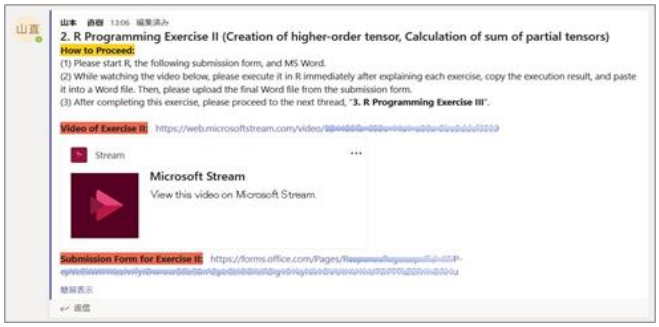

Fig. 8. Thread of Programming Exercise II.

As an example of a programming exercise, a problem using the 3D puzzle having a size of $3\times3\times3$ as shown in Fig. 2 was created. For positions where the light bulbs are set in the cube, the solution no.1 ({1,5,9,11,15,16,21,22,26}) shown in section II.A was used. This 3D puzzle is utilized from Programming Exercises II to IV. The example created here to perform the exercise is as follows.

[Example (Ex.) 2-1]

In the cube on the upper right of Fig. 9, the light bulbs are placed at the positions of the following elements.

{1,5,9,11,15,16,21,22,26}

At this time, all the positions appear to be illuminated when viewed from the three axes of the Cartesian coordinates. Create the third-order tensor represents this cube.

(End of example)

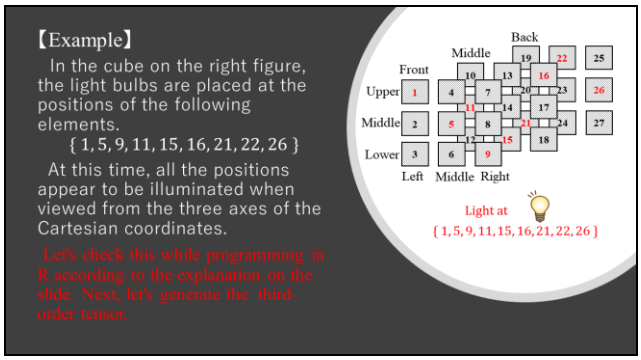

Fig. 9. Example using 3D puzzle.

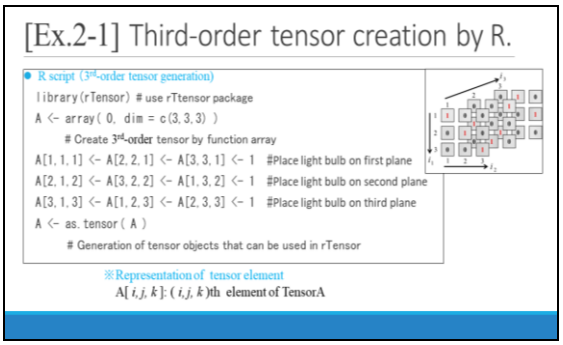

Fig. 10. Screen shot of explanatory video of programming for Ex.2-1.

Fig. 10 is a part of an explanatory video of a programming exercise for creating the third-order tensor of the cube shown in Fig. 9. Similarly to exercise I, while watching the explanation of the video, students can learn by programming in their own R environment, executing scripts, and checking whether the tensor is created correctly. They will make a report by copying scripts and execution results and pasting them into their MS Word file.

Then they perform a programming exercise of the sum of partial tensors while watching the video. A Word file of the completed report can be submitted from an assignment submission form created using a file upload function of MS Forms shown in Fig. 11. For more information on this figure, see "Question content of form for Exercise II" in the next section.

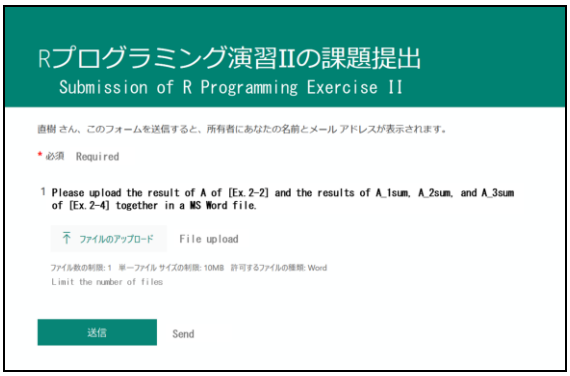

Fig. 11. Assignment submission form of Exercise II.

Fig. 12 shows a thread of Programming Exercise III mentioned in the previous section. In this exercise, students will perform the programming related to the matrix unfolding of higher-order tensors. This thread structure is the same as in Exercises I and II, and a method of video distribution and assignment submission is also the same as in Exercise II.

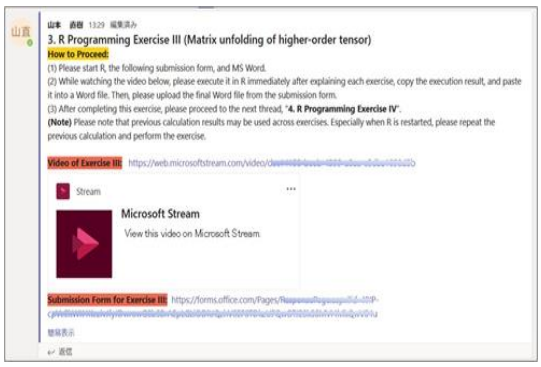

Fig. 12. Thread of Programming Exercise III.

In the exercise III, an example created using the 3D puzzle is as follows.

[Example (Ex.) 3-1]

In the cube on the upper right of Fig. 9, the light bulbs are placed at the positions of the following elements.

{1,5,9,11,15,16,21,22,26}

At this time, all the positions appear to be illuminated when viewed from the three axes of the Cartesian coordinates. Create a 2D map showing the positions where the light bulbs are set by using the 1-mode matrix unfolding.

(End of example)

Fig. 13 shows an explanation of the exercise in this example. This exercise is presented after an explanation of the matrix unfolding and specification of the unfold function of R were explained. At the bottom of the figure, the map of the positions where the light bulbs are set is displayed in the 1-mode matrix unfolding by the number 1. After showing this figure, the explanation how to read the map is given.

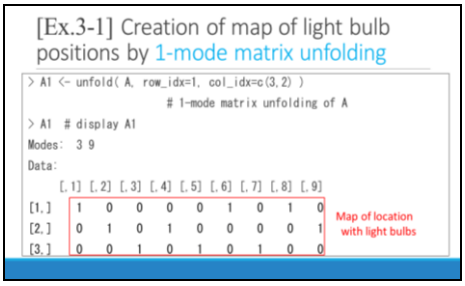

Fig. 13. Screen shot of explanatory video of programming for Ex.3-1.

Then students carry out programming exercises to create maps using the 2-mode and 3-mode matrix unfoldings and summarize them in a report by MS Word. When they complete the report, submit it by similar form as in Exercise II.

Fig. 14 is a thread of Programming Exercise IV described in the previous section. In this exercise, students learn to program about the folding into the higher-order tensor. Composition of this thread is the same as in Exercises I to III.

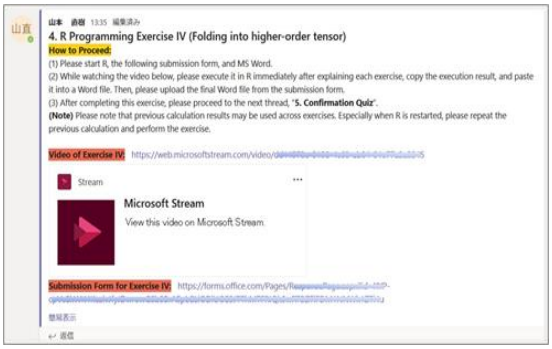

Fig. 14. Thread of Programming Exercise IV.

One of the examples created using the 3D puzzle in this exercise is shown below.

[Example (Ex.) 4-1]

Return the map of the placement of light bulbs expressed by the 1-mode matrix unfolding shown at the bottom of Fig. 13 to the original cube.

(End of example)

Fig. 15 is an explanatory part for an exercise of Ex. 4-1 above. This exercise is given after the specification of the fold function of R for folding into the higher-order tensor. By comparing the execution result and the image of folded third-order tensor, it can be confirmed that the original cube is restored as shown in the figure.

Then students perform programming exercises to fold the maps with the 2-mode and 3-mode matrix unfoldings into the original cubes and summarize results of them in a report. The completed report will be submitted using a designated form.

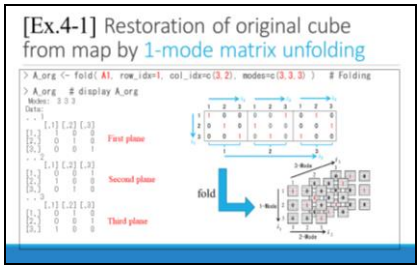

Fig. 15. Screen shot of explanatory video of programming for Ex.4-1.

Finally, students are asked to perform Confirmation Quiz shown in Fig. 16 to confirm whether they have learned the programming related to the higher-order tensor through the above exercises. In this figure, an explanation of how to proceed with the quiz is given in the first half of the thread, and a shared link is provided in the latter half for asking questions and submitting answers.

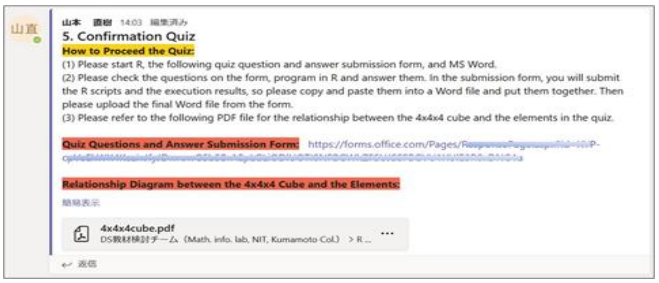

Fig. 16. Thread of confirmation quiz.

Fig. 17 shows a form created by using a quiz function and a file upload function of MS Forms. In this quiz, we set the following two questions.

[Question Content of the Confirmation Quiz Form]

Question 1. In a cube of size $4\times4\times4$, when light bulbs are set with the following combinations of element position numbers, select combinations in which all sides appear to be illuminated when viewed from three directions. Write an R program to solve this problem, run it and ask for the correct answer (multiple selections possible).

- ☐ 1, 2, 9, 14, 21, 23, 24, 25, 33, 35, 37, 41, 48, 49, 50, 51
- ☐ 1, 4, 7, 11, 13, 24, 26, 27, 29, 34, 41, 52, 53, 58, 59, 61
- ☐ 2, 7, 8, 9, 11, 12, 13, 18, 23, 24, 31, 38, 39, 44, 57, 61
- ☐ 1, 7, 12, 14, 20, 22, 25, 31, 34, 40, 43, 45, 51, 53, 58, 64

Question 2. Upload R scripts and output results of answers required in the following (1) to (3) in a MS Word file.

(1) An R script created to find the correct solution to the Question 1.

(2) For the combination obtained in question 1, create a map of positions where the light bulbs are set by using the 1-mode matrix unfolding. To create the map, make an R script and execute it. Then summarize this R script and the map obtained as the output result.

(3) Fold the map created in (2) into the original cube. To do this, create an R script and get the correct result. Then summarize this R script and resultant folded tensor.

(End of question content)

This Confirmation Quiz is created by using the solutions in the case of $m=4$ in the 3D puzzle described in section II.A. Fig. 18 shows the number assigned to each element of the $4\times4\times4$ cube, and the position where the light bulbs are placed are represented by that number. This figure is available from a shared link to the PDF file at the bottom of the thread shown in Fig. 16.

In the Question 1 of Fig. 17, four choices which are generated by the following algorithm is given.

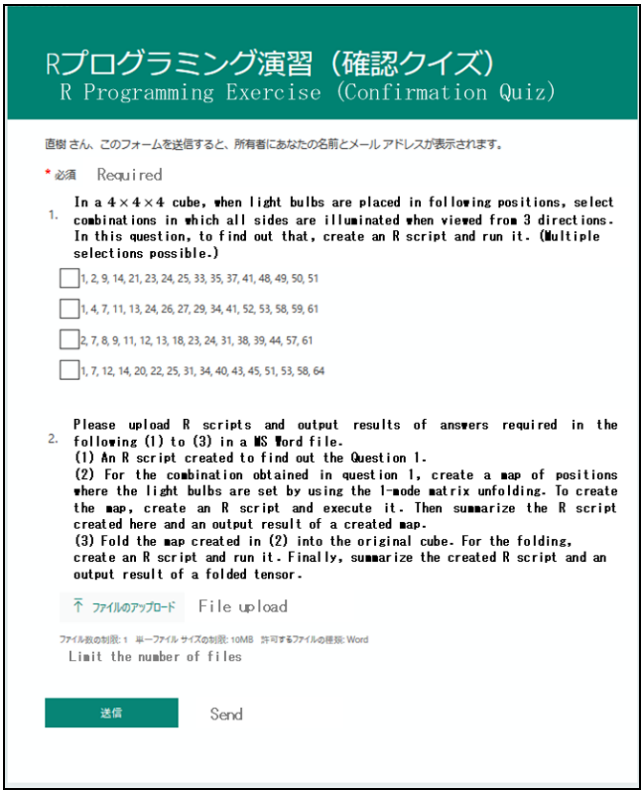

Fig. 17. Confirmation quiz form.

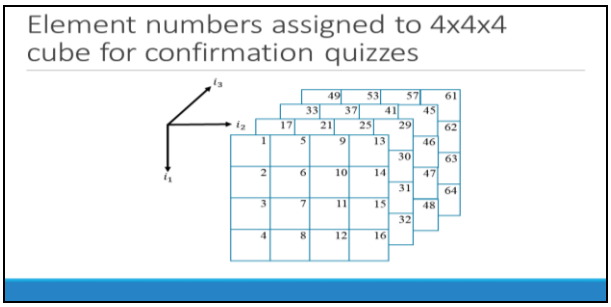

Fig. 18. Number assigned to each element of 4×4×4 cube.

[Algorithm for Choices Generation]

Input: 42 solutions of 3D puzzle in Table I.

Output: 4 choices of the Question 1 as shown in Fig. 17.

Step 1: Randomly extract a sequence from 42 solutions.

Step 2: Repeat 1) and 2) below until three sequences are obtained.

1) Randomly extract 16 numbers from 1 to 64 by non-restoring extraction. Then arrange those numbers in ascending order into a sequence.

2) If the sequence obtained in 1) does not exist in the 42 solutions, adopt it as a sequence of choices.

Step 3: Returns sequences of four choices obtained in Step 1 and Step 2.

(End of algorithm)

From this algorithm, it is found that the generated choices have one correct answer and three incorrect answers (however, if the axes of the cube are rotated, the incorrect answers may be correct ones). This algorithm was implemented by R and the output result was used in the form of MS Forms. Note that the choices for Question 1 are set so that multiple selections are possible, and the choices are shuffled when the form is started.

The Question 2 in Fig. 17 is about the R scripts and the

output results for each question. A learner summarizes them to a word file and submit using the file upload function of MS Forms.

IV. Trial Results of the Learning Material

We tried to the learning material described above for the students of our college, and the results are explained below. The participated students were three fifth-grade students (equivalent to second year of university) in information engineering-related department and two advanced course students (equivalent to fourth year of university). All of these students have experience with R programming, but the former students have no experience with rTensor package while advanced course students have.

First, the result of the questionnaire in the form of Fig. 7 will be described. The contents of the question are as follows.

[Question Content of the Form for Exercise I]

Question 1. Did you understand that the higher-order tensors dealt with here are multidimensional arrays?

Question 2. Did you understand the definition of higher-order tensors (multidimensional arrays)?

Question 3. In a higher-order tensor, the subscript of the tensor is called the mode and corresponds to the direction of the tensor. Did you understand the relationship between the subscript and mode of the third-order tensor and the direction of the tensor?

Question 4. Select all the exercises you have done.

☐ [Ex.1-1] rTensor Installation

(End of question)

Fig. 19 shows the response results for this form. On this form, Questions 1 to 3 are rated on a scale of 5, with 1 corresponding to “not understood”, 3 corresponding to “average understanding”, and 5 corresponding to “well understood”. It can be seen from Fig. 19 that the students can understand each content because the average score of all the questions is 4 points or more, and in particular, Question 1 has a maximum score of 5 points, so that they can understand it well.

Secondly, we will explain the results of the form of Exercise II in Fig. 11. This exercise is related to the generation of tensor and the calculation of tensor sum, and the questions are as follows.

[Question Content of the Form for Exercise II]

Question 1. Upload the result of A in [Ex.2-2] and the results of A_1sum, A_2sum, and A_3sum in [Ex.2-4] together in a MS Word file.

(End of question)

Looking at the submitted files, we found that most students combined the requested results with the script they created into a single file. In addition, it found that the execution result of [Ex.2-2] cannot be displayed when the old version of R was used.

Thirdly, the results of the form of Exercise III will be explained. This exercise is a programming exercise related to matrix unfolding, and the questions are as follows.

[Question Content of the Form for Exercise III]

Question 1. Upload the result of A1 in [Ex.3-1], A2 of [Ex.3-2], and A3 in [Ex.3-3] together in a MS Word file.

(End of question)

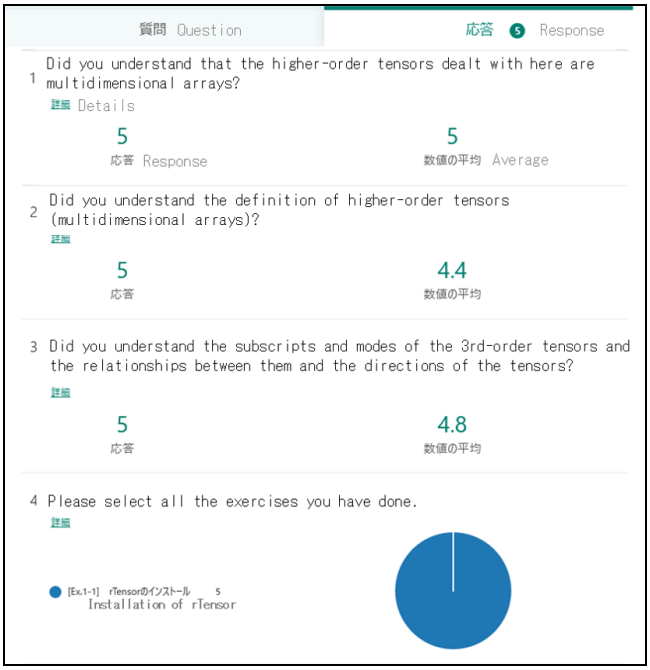

Fig. 19. Response results of the form for Exercise I.

From the submitted file, we saw that the students reported on the same stile as in Exercise II, and it was understood that all students were able to perform this exercise.

Fourthly, we will describe the results of the form of Exercise IV. This exercise is an exercise of folding the map represented by matrix unfolding into the original higher-order tensor. The questions in this form are as follows.

[Question Content of the Form for Exercise IV]

Question 1. Upload the three A_org’s that are the results for [Ex.4-1], [Ex.4-2], and [Ex.4-3] together in a MS Word file.

(End of question)

Upon checking the submitted file, three students were getting all correct output. Regarding the remaining two students, it was found that one student did not have the output result of [Ex.4-1], and the other student did not get the result because the version of R was old.

Finally, the result of the Confirmation Quiz form of Fig. 17 will be described. This quiz is to confirm whether the students have learned the programming related to higher-order tensor in the exercises so far. The detailed question contents of this form are shown in [Question Contents of the Confirmation Quiz Form] in the previous chapter.

Fig. 20 shows the results of the multiple-choice question of Question 1 in Confirmation Quiz. From this figure, it was seen that three out of five respondents answered correctly.

For Question 2, the submitted file showed that four students had the correct light bulbs placement and were able to create the scripts properly for all questions. From this, it was found that there was one student who answered Question 1 incorrectly but answered Question 2 correctly. Since the

options in Question 1 are shuffled each time the form is opened, it is thought that the change in the order from the first time when the form is opened the second time has an effect on it. The remaining one student chose the incorrect bulb placement in Question 1 but was able to create the script for that placement.

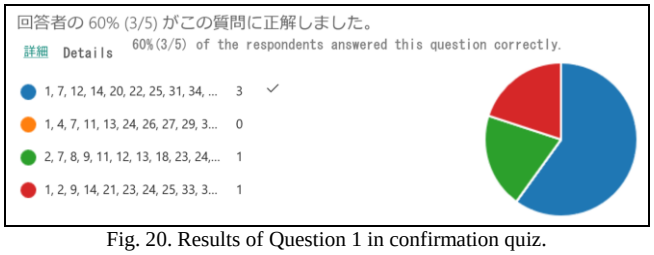

Fig. 20. Results of Question 1 in confirmation quiz.

I. Conclusion

In this research, we have developed learning material that assumes programming exercises related to higher-order tensors in data science subjects. A 3D puzzle was taken as learning material, expressed by a higher-order tensor, and the solution obtained by computer calculation was used as an example of practice tasks and application problems of confirmation quiz. The contents of the learning material are programming exercises of R for higher-order tensors, and cover tensor generation, partial tensor sum calculation, matrix unfolding, and folding.

This learning material is organized in a channel on MS Teams, so students can do programming exercises remotely. On the other hand, the teacher can check the learning status and evaluate the grade by looking at the questionnaire answers and the exercise report.

As a result of having five students try this learning material, all of them were able to execute Exercise I and Exercise III correctly. Exercises II and IV were carried out correctly except for a few cases that could not be done because the version of R was old. Regarding the confirmation quiz, the majority of the students were able to answer correctly, and the programming was correctly created for the remaining students. From the above results, it is considered effective to use remotely this learning material for programming exercises in data science subjects.

A future task is to actually incorporate them into data science-based subjects and to practice high-order tensors. Furthermore, we would like to add an exercise to calculate the product of higher-order tensor and matrix called n-mode product to this material and use it for learning tensor decomposition.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Naoki Yamamoto created most of the learning material and drafted the paper; Akio Ishida took responsibility in the construction of the manuscript; Kazuki Ogitsuka sought solutions of 3D puzzle; Nobuhiro Oishi reviewed the article; Jun Murakami organized and supervised the research.

References

[1] L. Eldén, "Matrix methods in data mining and pattern recognition," *SIAM*, 2007, ch. 8 and ch. 14.

[2] A. Ishida, N. Yamamoto, J. Murakami, and N. Oishi, "Solving 3-D puzzles using tensor decomposition and application to education of multidimensional data analysis," *International Journal of Machine Learning and Computing*, vol. 8, no. 5, pp. 447-453, October 2018.

[3] N. Yamamoto, J. Murakami, and A. Ishida, "Using 3D puzzles to help understand tensor decomposition programming," in *Proc. the Annual Software Symposium in Kumamoto*, pp. 115-124, June 2019.

[4] N. Yamamoto, A. Ishida, N. Oishi, and J. Murakami, "Development of teaching tool for supporting understanding of tensor decomposition using MacMahon's coloured cubes," *International Journal of Information and Education Technology*, vol. 10, no. 1, pp. 14-19, January 2020.

[5] R. C. Team, "R: A language and environment for statistical computing," *R Foundation for Statistical Computing*, Vienna, Austria.

[6] T. Nishiyama, "On some articles of vol. 86," *SHOTOH SUGAKU*, vol. 87, pp. 60-63, 2019.

[7] A. Ishida, K. Ogitsuka, N. Yamamoto, N. Oishi, and J. Murakami, "On the 4×4×4 cube puzzle on p. 60 of Vol. 87 (the second part)," *SHOTOH SUGAKU*.

[8] A. Ishida, K. Ogitsuka, N. Yamamoto, N. Oishi, and J. Murakami, "On the 4×4×4 cube puzzle on p. 60 of Vol. 87 (the first part)," *SHOTOH SUGAKU*.

[9] L. Lathauwer, B. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253-1278, 2000.

[10] J. Li, J. Bien, and M. Wells. (March 2020). Package 'rTensor'. [Online]. Available: https://cran.r-project.org/web/packages/rTensor/rTensor.pdf

Copyright © 2021 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).

N. Yamamoto received the Ph.D. in engineering from Kyushu Institute of Technology, Japan, in 2001. He is currently a professor of the Department of Human-Oriented Information Systems Engineering, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of multidimensional data analysis and numerical calculation. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and Kyushu Society for Engineering Education.

A. Ishida received the M.S. and Ph.D. in science from Kumamoto University, Japan, in 2010 and 2014. He is currently an assistant professor of the Faculty of Liberal Arts, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of multi-dimensional data analysis and numerical calculation.

K. Ogitsuka is currently an undergraduate student of the Department of Human-Oriented Information Systems Engineering, Kumamoto College, National Institute of Technology, Japan. After graduating, he will transfer to the Faculty of Computer Science and Engineering at Toyohashi University of Technology, Japan. His research interests are in the area of numerical analysis.

N. Oishi received the M.E. from Toyohashi University of Technology, Japan, in 1988 and Ph.D. in engineering from Kyushu Institute of Technology, Japan, in 2004. He is currently a professor and the dean of the Department of Information, Communication and Electronic Engineering, Kumamoto College, National Institute of Technology, Japan. His research area of interests includes numerical calculation, statistical analysis, and causal analysis of multidimensional data. He is a member of the Physical Society of Japan (JPS) and the Japan Society of Applied Physics (JSAP).currently an undergraduate student of the Department of

Human-Oriented Information Systems Engineering, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of numerical analysis.

J. Murakami received the Ph.D. in engineering from Toyohashi University of Technology, Japan, in 2000. He is currently a professor of the Department of Human-Oriented Information Systems Engineering, the Director of Library, Kumamoto College, National Institute of Technology, Japan. He is also a Board Member and Planning Chairman of the Research for Innovation and Synthesis of Technology in Kumamoto. His research area of interests includes statistical analysis, numerical calculation, and digital signal processing. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), and the Human Interface Society Japan (HISJ).