Part IV

Mahalanobis–Taguchi System (MTS)

Human Performance (Cases 67–68)
Inspection (Cases 69–73)
Medical Diagnosis (Cases 74–78)
Product (Cases 79–80)

Taguchi's Quality Engineering Handbook. Genichi Taguchi, Subir Chowdhury and Yuin Wu

CASE 67

Prediction of Programming Ability from a Questionnaire Using the MTS

Abstract: In this research, by taking advantage of the Mahalanobis–Taguchi system (MTS), we evaluated the ability of respondents to write software, based on their answers on questionnaires.

1. Objective of Experiment

To collect the necessary data, we asked 83 testees to answer a questionnaire and create a program. The 83 testees broadly covered programmers, businesspeople, undergraduate students, and college students, all with some knowledge of programming. The questionnaire consisted of 56 questions related to programming and each testee was asked to respond from the following seven-point scale: "positively no (−3)," "no (−2)," "somewhat no (−1)," "yes or no (0)," "somewhat yes (+1)," "yes (+2)," and "positively yes (+3)." The program that each testee worked on was a simple mask calculation (BASE + BALL = GAMES: each alphabet corresponds to a single digit and the same alphabet needs to have the same digit). That is, we believed the results would show that people who were able to write such a program would answer in a certain way. Our model is shown in Figure 1.

2. Calculation of Base Space

After checking over all the programs that the testees created, we defined those who created an easily readable program without bugs as "persons with a sense of programming." Among the 83 testees there were only 17 persons with a sense of programming. Using the remaining 66 persons with no sense of programming, we formed a base space. The total number of items was 60.

Figure 2 shows a histogram of Mahalanobis distances for all data. Give an appropriate threshold, we can completely separate "persons with a sense of programming" and "those with no sense of programming." According to this result, we concluded that a Mahalanobis distance can be applied to questionnaire data.

However, when we judged whether or not a person has a sense of programming, the probability of making a correct judgment is significant. That is, we need to ascertain the reliability of the base space that we have formed.

Provided that we have numerous data, for example, 300 normal data, using 285 normal data to create a base space, we compute the distance from the base space for each of the remaining 15 data. As long as the distances of these 15 sets of data are distributed around 1, the base space created turns out to be reliable. If the distance becomes larger, we conclude that the number of data needed to construct the base space is lacking or that the items selected are inappropriate.

Yet, due to the insufficient number of data for creation of the base space in this study, even if only five pieces of data are removed from the base space, the reliability of the space tends to be lowered dramatically. Therefore, by excluding only one data point from the base space, we created a base space without the one point excluded and calculated the distance of the point excluded. As a next step, we removed another data point and restored the point excluded previously in the base space. Again, creating a new base space without one piece of data,

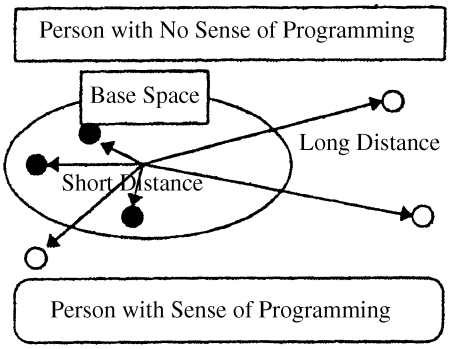


Figure 1
Base space for programming ability

we computed the distance for the data currently removed. By reiterating this process, we evaluated the reliability of the standard space.

Figure 3 shows the results obtained using the reliability evaluation method described above. The result reveals that the current base space cannot be used for a judgment because "persons with no sense of programming" is also distant from the base space. Such a base space cannot be used for judgment, primarily because of the small number of data.

3. Selection of Items

There are some reasons that the distance for "a person with no sense of programming" resulted in a

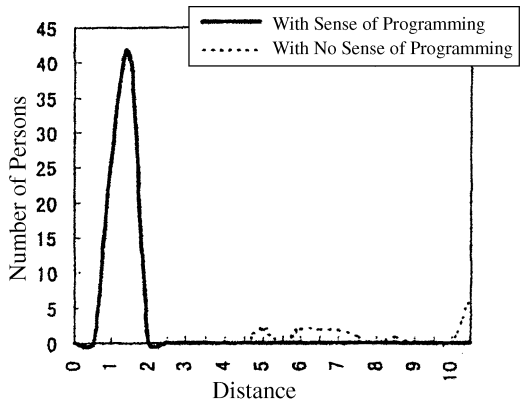


Figure 2
Distances from base space for "persons with no sense of programming"

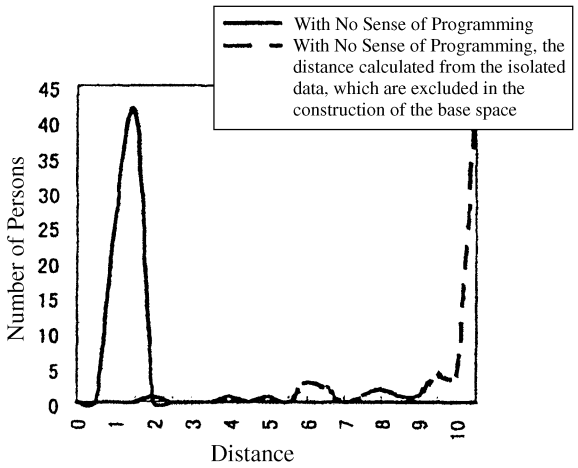


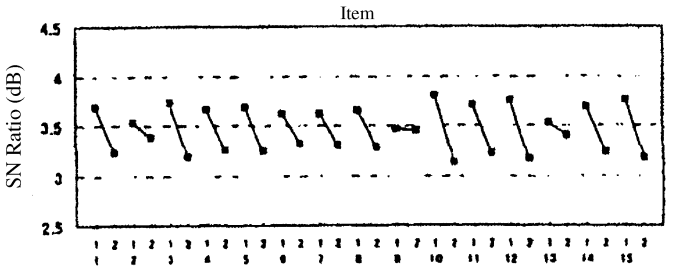
Figure 3
Data in base space for reliability evaluation

large value. When there are many items in a base space, unnecessary items for a judgment are sometimes included in the space. By identifying these items, we attempted to exclude them from the base space.

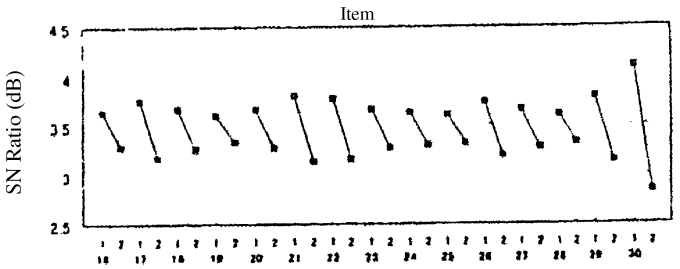
We allocated each item as a factor to an orthogonal array with two levels. Since we now had 60 items, an L_{64} orthogonal array was selected. For each item, level 1 was used to create a base space. Without using level 2 for the base space, from each row in the L_{64} orthogonal array, we formed one base space for "persons no sense of programming." For the resulting base space, we computed distances for abnormal data or "persons with a sense of programming." Since these abnormal data should be distant from the base space, we utilized a larger-the-better SN ratio.

$$\eta = -10 \log \frac{1}{n} \left(\sum_{i=1}^{n} \frac{1}{D_i^2} \right) \tag{1}$$

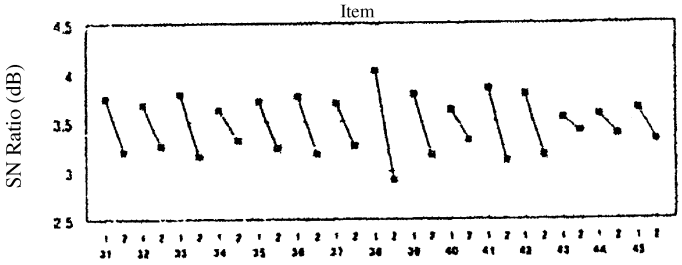
Based on the response graphs, we checked over effective and ineffective items to decide whether a certain person had a sense or no sense of programming. The items at level 1 were used to create a base space. Now, because the items at level 2 were not used for the base space, an item with a declining tendency from left to right was considered effective for a judgment, whereas an item with a contrary trend was regarded to affect judgment negatively.



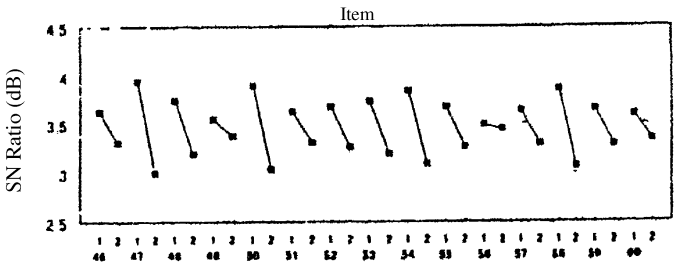
(a) Response Graphs for Item Selection (Questions 1–15)



(b) Response Graphs for Item Selection (Questions 16–30)



(c) Response Graphs for Item Selection (Questions 31–45)



(d) Response Graphs for Item Selection (Questions 45–60)

Figure 4
Response graphs for item selection

Figure 4 demonstrates that all of the items affected a judgment positively. Now, by eliminating the least effective item for question 9, we selected effective items once again. In this second item selection, only question 13 showed an increasing trend from left to right. Then, excluding question 13, we returned question 9 to the base space because it did not have an increasing trend but a small effect (in fact, question 9 will be removed in the later item selection because it did show a rising trend from left to right).

In the third and later item selections, every time we had at least one item with a rising trend from left to right we eliminated it (or them) from the base space and selected other items. Nevertheless, once the total number of items was reduced to about 50, all of the items turned out to be effective or with a decreasing trend from left to right once again. At this point in time, fixing the 20 items, the larger ones, and keeping them fixed without assigning them to an orthogonal array, but assigning the remaining 30 data to an L_{32} orthogonal array, we selected effective items. In sum, by changing an orthogonal array used, we attempted to seek items that showed poor reproducibility. Through this process, we expected to find only high-reproducibility items. Nevertheless, since we removed items, even if all the items indicated were more or less effective, the distance calculated from abnormal data was reduced. In terms of the SN ratio, the reduction in the number of items leads to a smaller SN ratio. However, since as compared to this diminution of the SN ratio, the average distance outside the base space (with no sense) approximates that in the base space, we concluded that this analysis procedure was improved as a whole.

Through this procedure, we reduced the number of items from 60 to 33 as shown in Figure 5. By doing so, "persons with no sense of programming" who were excluded from the base space started to overlap 25% of the base space. To make them overlap more, we needed to increase the data. The contents of the questionnaire are abstracted in the list below. All of the questions were composed of items that some software- or programming-related books regard as important for programmers and were considered essential from our empirical standpoint. The list includes only items that have high reproducibility and are associated with a sense of programming.

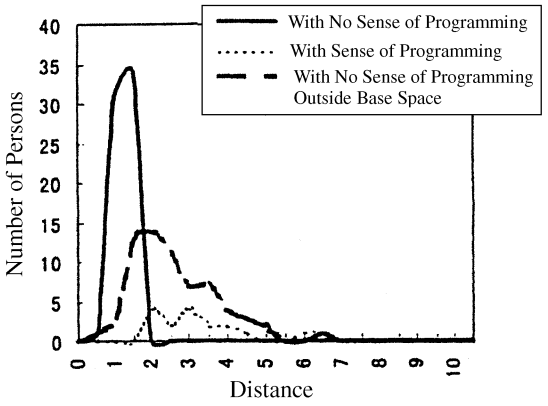


Figure 5
After-item-selection data

1. Do you tend to be distracted from your job if a certain job's result will come out after one week?
2. Do you mind your appearance?
3. Do you often clean up things around you (belongings or files in a hard disk)?
4. Do you tend to be absorbed without minding time while you are doing favorite things?
5. When everyday jobs change drastically, do you tend to feel happy about responding to them?
10. Do you like puzzles (jigsaw puzzles, Rubik's cube, or puzzle rings)?
12. Do like to use a computer (personal computer or workstation)?
14. Do you like to speak publicly?
15. Do you tend to be superstitious when unfavorable things happen?
16. Do you like any type of programming regardless of its contents?
17. Do you often play TV games?
19. Are you in the forefront of fashion?
20. Do you like to explain with a chart when asked by others?
21. Do you have beliefs?
22. Do you like to handcraft something?

Table 1
Average and standard deviation for each effective question

	With No Sense		With Sense		Difference between Averages
Question	Average	Standard Deviation	Average	Standard Deviation	
1	0.17	1.61	0.18	1.51	0.01
2	0.67	1.42	0.29	1.40	0.37
3	−0.05	1.55	0.12	1.76	0.16
4	2.12	0.92	1.76	1.09	0.36
5	0.41	1.53	−0.35	1.11	0.76
10	0.58	1.56	0.65	1.90	0.07
12	1.29	1.41	1.53	1.37	0.24
14	−0.47	1.60	−1.24	1.39	0.77
16	0.32	1.63	1.35	0.86	1.03
17	0.32	1.97	−0.06	2.14	0.38
19	−0.24	1.48	−0.18	1.55	0.07
20	0.53	1.25	0.29	1.53	0.24
21	0.77	1.41	0.40	1.22	0.37
22	1.44	1.45	1.14	1.50	0.30
23	0.85	1.42	0.29	1.36	0.55
24	0.41	1.93	0.53	1.81	0.12
26	0.39	1.50	0.76	1.52	0.37
27	−0.14	1.82	0.41	1.87	0.55
30	−0.08	1.71	−0.41	1.97	0.34
31	0.31	1.76	0.29	1.76	0.02
35	−1.09	1.57	−0.76	1.68	0.33
37	1.33	1.40	1.18	1.29	0.16
38	−0.03	1.69	−0.41	1.28	0.38
39	0.65	1.45	0.29	1.45	0.36
41	0.33	1.27	0.35	1.54	0.02
47	0.75	1.19	0.24	1.52	0.52
50	0.67	1.14	0.29	1.05	0.37
51	−0.63	1.77	0.53	1.87	1.15
54	0.19	1.87	−0.53	1.94	0.72
55	0.19	1.58	0.18	1.38	0.01
56	−0.16	1.60	−0.06	1.78	0.10
58	1.12	0.33	1.00	0.00	0.12

Table 2
Distances from base space using data classified by type of program

Class	Loop Type	Logic Type and Other	Outside Loop Type
0	0	0	0
0.5	0	0	0
1	28	0	0
1.5	38	0	0
2	0	0	0
2.5	0	0	0
3	0	0	1
3.5	0	1	0
4	0	0	0
4.5	0	0	0
5	0	0	1
5.5	0	0	1
6	0	0	1
6.5	0	0	1
More than 7	0	16	61
Total	66	17	66

Table 3
Distances from base space using after-item-selection data classified by type of program

Class	Loop Type	Logic Type and Other	Outside Loop Type
0	0	0	0
0.5	0	0	0
1	37	0	0
1.5	29	2	3
2	0	0	4
2.5	0	3	9
3	0	1	5
3.5	0	6	16
4	0	1	2
4.5	0	1	6
5	0	0	4
5.5	0	0	3
6	0	0	1
6.5	0	1	4
More than 7	0	2	9
Total	66	17	66

23. Are you worried about things that have gone wrong with you?
24. Are you willing to watch TV sports programs?
26. Do you like to read specialized books?
27. Can you continue to do simple work with patience?
30. Do you like to draw pictures?
31. Do you tend to be annoyed by your surroundings while pondering something?
35. Do you sometimes forget to button up your clothes?
37. Are you curious about the mechanisms of machines?
38. Do you mind if your name is spoken incorrectly?
39. Do you tend to pursue perfection in your job?
41. Do you tend to take action as soon as you make up your mind?
47. Do you feel that you are different from others?
50. Do you tend to think about things that are coming up when doing a job?
51. Can you wake up early in the morning?
54. Do you tend to get lost when visiting an unknown town?
55. Do you make more mistakes in spelling or in filling out a form?
56. Do you dislike having to socialize with others, for example, at a party?
58. Gender

Table 1 shows the distribution of the data for "persons with a sense of programming" and "those with no sense of programming" in the question-

naire. However, for any question, there is no significant difference in sense of programming. In other words, we cannot make a judgment by using only individual questions. Once a certain number of questions are grouped, we distinguished "persons with a sense of programming" from "those with no sense of programming."

4. Base Space Classified by Type of Program

We attempted to create a base space using another classification, categorizing the data by a type of program. The programs that we used as a sample can be roughly classified into three types. *Loop type* is a program that changes numbers from 0 to 9 in a loop process and searches for an answer on a trial-and-error basis in the mask calculation. In this research, the largest number of testees, 66, took up this method. Despite its wide applicability, it takes a lot of time to arrive at the answer. *Logic type* finds out an answer according to the logic that "if A + B = C, then. . . ." Although this method is not broadly applicable, we can use it to arrive at an answer quickly. Finally, *other* is a category for complicated programs that cannot be classified in the aforementioned two groups. In fact, some are excellent programs, whereas some do not take shape as a program.

Assuming that "if a program is written based on the same idea as those of other persons, they can also easily debug the program," by using the data for loop type we created a base space to seek a group with the same way of thinking in programming. The total number of data in the base space turned out to be identical to that in the case of classifying the data by a sense of programming.

Table 2 summarizes the histogram of the before-item-selection data (Figure 5), and Table 3 shows that of the after-item-selection data. Even after the items were selected, no judging ability can be seen in the base space based on the classification by type of program. This fact demonstrates that even using data from the same questionnaire, if we opt to use a different setup of a base space or a different classification of data, we can never make a judgment. In sum, the initial classification is regarded as the most crucial.

References

Kei Takada, Muneo Takahasi, Narushi Yamanouchi, and Hiroshi Yano, 1997. The study on evaluation of capability and errors in programming. *Quality Engineering*, Vol. 5, No. 6, pp. 38–44.

Kei Takada, Kazuhito Takahashi, and Hiroshi Yano, 1999. The study of reappearance of experiment in evaluation of programmer's ability. *Quality Engineering*, Vol. 6, No. 1, pp. 39–47.

Kei Takada, Kazuhito Takahashi, and Hiroshi Yano, 1999. A prediction on ability of programming from questionnaire using MTS method. *Quality Engineering*, Vol. 7, No. 1, pp. 65–72.

Muneo Takahashi, Kei Takada, Narushi Yamanouchi, and Hiroshi Yano, 1996. A basic study on the evaluation of software error detecting capability. *Quality Engineering*, Vol. 4, No. 3, pp. 45–53.

This case study is contributed by Kei Takada, Kazuhito Takahashi, and Hiroshi Yano.