International Journal of Computational and Theoretical Statistics
ISSN (2384-4795)
Int. J. Comp. Theo. Stat. 6, No. 1 (May-2019)

http://dx.doi.org/10.12785/ijcts/060104

Naive Principal Component Analysis in Software Reliability Studies

Loganathan A[1] and R Jeromia Muthuraj[2]

[1,2] *Department of Statistics, Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu, India.*

Received August 2, 2018 Revised January 17, 2019, Accepted February 18, 2019, Published May 1, 2019

Abstract: Software usage has been dealing major parts in all the activities of individuals as well as organizations. Software users expecting the good and reliable software. There are many approaches in Software reliability studies probabilistic and non-probabilistic approaches. Zhang and Pham (2000) defined third two environmental factors for studying the reliability of software and categorized them into five groups. Later they proposed to use information about three principal components extracted from ten environmental factors. It causes loss of information about the remaining twenty-two factors, two more environmental factors have been recommended as significant factors in a subsequent literature for studying the reliability of software. This paper proposes a methodology to use the information about all the thirty-four factors through principal components reducing the volume of information with less amount of loss of information. Information gained from the different stages of PCs is compared with Shannon Information measure.

Keywords: Software Reliability, Environmental Factor, Clustering, Principal Components, Shannon Entropy.

1. Introduction

Software reliability is the probability that a software will function without failure under a given environmental condition during a specified period of time. Software Reliability Modeling plays a vital role in developing software systems and enhancing computer software. Software reliability theory deals with probabilistic methods applied to the analysis of random occurrence of failures in a given software system. A software is said to contain a fault if, for input data, the output result is incorrect. Fault is always an inevitable part in software codes. Therefore, the process of software debugging is a fundamental task of the life cycle of a software system. Software has become a necessary part of industry, medical systems, spacecraft and military systems, commercial systems and all the practical applications. So the reliability of the software is very essential. Software reliability is a measure of the quality and performance of a software package. From the statistical point of view, software reliability deals with probabilistic methods applied to the analysis of random occurrences of failures in a software system. There are many hardware reliability approaches but Software Reliability Modeling (SRM) work started in the early '70s, with the inventive works of Jelinski and Moranda (1972), Shooman and Coutinho. After that many works were done related to software reliability. Many software reliability models were constructed in parametric and non-parametric approaches. Some parametric models are Jelinski and Moranda De-Eutrophication Model (1972), Schick and Wolver ton Model, Goel and Okumoto Imperfect Debugging Model, Littlewood - Verrall Bayesian Model (1973), Goel-Okumoto Non homogeneous Poisson Process Model, Shooman Exponential Model, and etc. Some Non Parametric models are A Non-Parametric Order Statistics Software Reliability Model (1998), State Transition Model for Predicting Software Reliability (2007), and etc. The experts say that there are more than 225 software reliability models. But there is not even a single model that can be used in all situations. A model may work well for a set of certain software, but it may be completely off track for other kinds of problems.

Zhang and Pham (2000) pointed out that consideration of information about such environmental factors in the construction of software reliability models would be more meaningful. In this context, they proposed a set of 32 environmental factors arguing that information about such factors will be more relevant to study software reliability.

E-mail address: jeromstat@gmail.com , aln_24@rediffmail.com

Patwa and Malviya (2014) also proposed a set of 26 factors recommending them as potential environmental factors. Among them, 24 factors exist in the list of factors proposed by Zhang and Pham (2000). Thus, there are 34 potential environmental factors which can influence the quality level of the software. Since there is a correlation structure among the environmental factors, it would be difficult to construct a software reliability model with uncorrelated factors.

Zhu et. al., (2015) recommended to consider three principal components (PCs) extracted from ten environmental factors selected according to their ranks. It will increase the loss of information about all the environmental factors.

Loganathan and Jeromia Muthuraj (2016) proposes a new methodology for data reduction using principal component analysis. It will helps to decrease the loss of information in the environmental factors.

This paper attempts to determine the Naive Principal Component Analysis by clustering the 34 environmental factors using hieratical clustering procedure Euclidean Distance. Section 2 describes the methodology for collecting the information about the environmental factors from software engineers. Number of Clusters from the 34 environmental factors are presented in section 3. Within cluster PCs, between cluster PCs compared with over all PCs from the 34 factors by Shannon Information Measure in section 4. Results are summarized in section 5.

2. Data on Environmental Factors

Zhang and Pham (2000) introduced thirty two factors which is important for any software to find reliability and given a name as "Environmental Factors". Zhang and Pham (2000) and Zhu et. al.,(2015)grouped the environmental factors into five phases. Patwa and Malviya (2014) proposed 26 factors as potential environmental factors to assess the reliability of software. Among them, two factors can be considered as new factors and the remaining twenty four factors are among the list of thirty two factors presented thirty two variables.

34 Environmental factors are listed in Table 1.

Table I Environmental Factors and Their categorization

Factor Number	Category	Environmental Factor
	General	
F01		Program Complexity
F02		Program Categories
F03		Difficulty of Programming
F04		Amount of Programming effort
F05		Level of Programming technologies
F06		Percentage of Reused modules
F07		Programming Language
F08		Complexity in Logic
	Analysis and Design	
F09		Frequency of Program specification change
F10		Volume of Program design documents
F11		Design Methodology
F1		Requirements Analysis
F13		Relationship of detailed Design to Requirement
F14		Work Standards
F15		Development Management
	Coding	
F16		Programmer Skill
F17		Programmer Organization
F18		Development Team size
F19		Program Workload (stress)
F20		Domain Knowledge
F21		Human Nature
	Testing	
F22		Testing Environment

F23		Testing Effort
F24		Testing Resource allocation
F25		Testing Methodologies
F26		Testing Coverage
F27		Testing Tools
F28		Documentation
	Hardware systems	
F29		Processors
F30		Storage Devices
F31		Input/output Devices
F32		Telecommunication Devices
F33		System Software
F34		Random Access Memory

Since all the 34 factors have potential to study software reliability, it is proposed that the 34 environmental factors shall be used for software reliability assessment. Even though the factors are listed in Tables 1 and 2 under different phases, it may be expected that the factors within phases may be dissimilar and between phases may be similar.

Since all the 34 factors are essential, none of them shall be eliminated. Ranking of the factors may not have any meaning, but the factors with similar importance may be grouped together. Information about each factor may be considered for analysis.

For this purpose, opinions about the relevance of all the 34 factors were invited from 25 randomly selected respondents. They are software developers in organizations of various kinds such as commercial, web-designing and inside-user organizations. The respondents expressed their opinion about the level of significance of each factor with scores ranging from 0 through 7. The score 7 represents "Extremely Significant", 6 represents "More Significant", 5 represents "Moderately Significant", 4 represents "Significant", 3 represents "May and May not Significant", 2 represents "Less Insignificant", 1 represents "Moderately Insignificant" and 0 represents "Not Significant". Some of the respondents expressed their opinion for some factors with the score of "3" mentioning that the level of significance of the factors is software dependent.

3. CLUSTERING OF ENVIRONMENTAL FACTORS

Similarities among the factors are studied applying the hieratical clustering procedure Euclidean Distance single linkage nearest neighbor method, upon the scores assigned to the factors (Kaufman and Rousseeu (1990)). The dendrogram is displayed in Figure 1.The dendrogram shows that similarity among the 34 environmental factors forms 6 homogenous clusters. The factor are in this clusters are similar and between clusters are dissimilar.

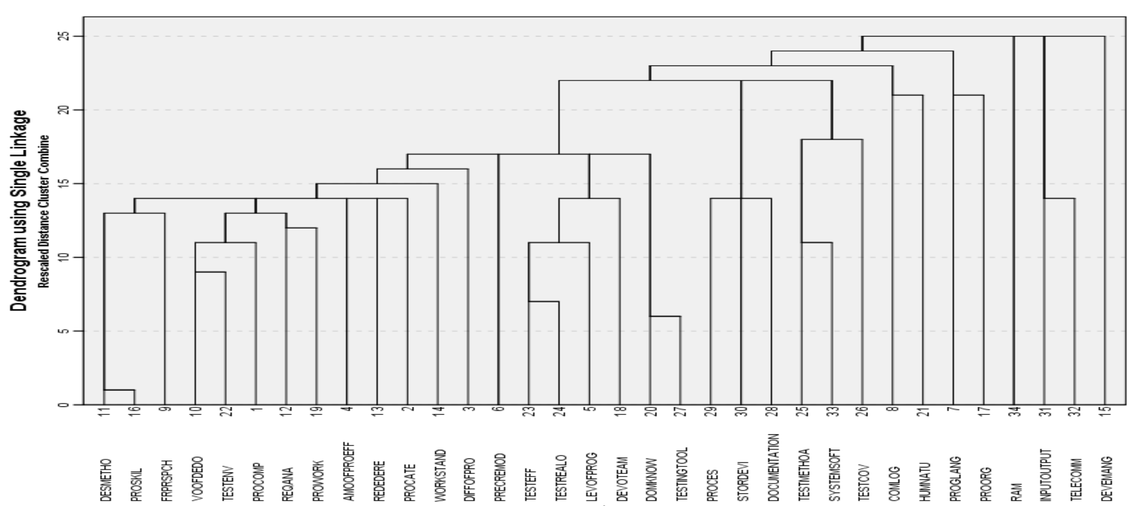

Figure 1. Dendrogram

Table II Cluster of Environmental Factors

Cluster	Factor Number	Factors
Cluster 1	F11	Design Methodology
	F16	Programmer Skill
	F9	Frequency of Program specification change
	F10	Volume of Program design documents
	F22	Testing Environment
	F1	Program Complexity
Cluster 2	F12	Requirements Analysis
	F19	Program Workload (stress)
	F4	Amount of Programming effort
	F13	Relationship of detailed Design to Requirement
	F2	Program Categories
	F14	Work Standards
	F3	Difficulty of Programming
Cluster 3	F6	Percentage of Reused modules
	F23	Testing Effort
	F24	Testing Resource allocation
	F5	Level of Programming technologies
	F18	Development Team size
	F20	Domain Knowledge
	F27	Testing Tools
Cluster 4	F29	Storage Devices
	F30	Human Nature
	F28	Documentation
	F25	Testing Methodologies
	F33	System Software
	F26	Testing Coverage
Cluster 5	F8	Complexity in Logic
	F21	Processors
	F7	Programming Language
	F17	Programmer Organization
Cluster 6	F34	Random Access Memory
	F31	Input/ Output Devices
	F32	Telecommunication Devices
	F15	Development Management

4. CLUSTERING OF ENVIRONMENTAL FACTORS

Pham (2000), Zhang and Pham (2000), Zhu et. al., (2015), Patwa and Malviya (2014) studied the existence of relationship among the environmental factors using Karl Pearson’s formula. Though the use of Karl Pearson’s formula was not justified, it may be noted that the environmental factors are correlated. In some statistical analysis, the variable/factors under investigation should be uncorrelated. Principal component analysis extracts uncorrelated linear combinations of the variables/factors under investigation (Jollife (2005)). Results of PC analysis also provide information about proportion of total variation in the data explained by each PC. Accordingly, desirable number of PCs may be selected from the order of proportions of variation. Here, it is proposed to select PCs within each cluster so that the selected PCs in each cluster explain, in total, of 90% of total variation in the scores assigned to the environmental factors within the cluster. The selected PCs in each cluster are presented in Table III.

Table III (Principal Components of Environmental Factors)

Cluster	PC	Factors and Their Co-efficients							Cumulative % of Variation
Cluster 1		F11	F16	F9	F10	F22	F1	-	
	CPC_{11}	0.777	0.838	0.524	0.055	0.644	0.743	-	38.97%
	CPC_{12}	0.208	-0.134	-0.616	0.847	0.353	0.188	-	64.60%
	CPC_{13}	0.33	-0.152	0.406	0.427	-0.567	0.575	-	80.60%
	CPC_{14}	-0.374	-0.174	0.423	0.241	0.313	-0.143	-	90.69%
Cluster 2		F12	F19	F4	F13	F2	F14	F3	
	CPC_{21}	0.819	-0.799	-0.511	-0.194	0.076	0.735	0.622	35.92%
	CPC_{22}	0.316	0.297	-0.49	0.743	-0.649	-0.107	-0.015	59.48%
	CPC_{23}	-0.191	-0.166	0.611	0.188	-0.507	0.56	-0.603	77.86%

	CPC_{24}	0.157	0.033	0.234	0.575	0.555	0.118	0.458	90.08%
Cluster 3		F6	F23	F24	F5	F18	F20	F27	
	CPC_{31}	0.671	0.564	0.622	0.659	0.078	-0.015	0.859	39.92%
	CPC_{32}	-0.273	0.55	-0.015	-0.274	0.816	-0.603	-0.063	69.26%
	CPC_{33}	0.376	-0.305	-0.603	0.394	0.453	0.452	-0.027	91.43%
Cluster 4		F29	F30	F28	F25	F33	F26	-	
	CPC_{41}	-0.273	0.803	-0.147	0.876	-0.101	0.768	-	30.33%
	CPC_{42}	0.315	0.388	0.457	-0.086	0.821	-0.019	-	53.16%
	CPC_{43}	0.727	0.101	-0.695	0.024	0.062	0.497	-	73.67%
	CPC_{44}	0.54	0.07	0.533	0.134	-0.523	-0.680	-	91.11%
Cluster 5		F8	F21	F7	F17	-	-	-	
	CPC_{51}	0.394	-0.248	0.859	0.768	-	-	-	38.61%
	CPC_{52}	0.677	0.802	-0.063	-0.019	-	-	-	70.25%
	CPC_{53}	-0.589	0.508	-0.027	0.497	-	-	-	90.57%
Cluster 6		F34	F31	F32	F15	-	-	-	
	CPC_{61}	-0.387	0.868	0.499	-0.507	-	-	-	35.25%
	CPC_{62}	0.797	0.004	0.764	0.15	-	-	-	66.26%
	CPC_{63}	-0.3	0.278	0.145	0.847	-	-	-	91.89%

5. Shannon Information Measure

Entropy is the average amount of the information from the event. This entropy is introduced by Shannon in 1948, in the seminal papers in the field of information theory. It defined, information strictly in terms of the probabilities of events.

Therefore, let us suppose that we have a set of probabilities

$$P = \{p_1, p_2, ...p_n\}$$

Then the entropy of the distribution P by:

$$H(P) = \sum_{i=1}^{n} p_i * \log(1/p_i)$$

If it is a continuous rather than discrete probability distribution P(x) then:

$$H(P) = \int p(x) * \log(1/p(x))dx$$

Here from all the Principal Components from thirty four factors and the Dual Principal Components from 23 PCs are compared with Shannon Information Measure. The average amount of information in gained by Principal Components from all the thirty four factors is 15.69. The average amount of information in gained by liner combination of Naive Principal Components the thirty four factors is 17.02.

6. Summary

This study considered 34 potential environmental factors which are important to study reliability of the software. Data were collected from software developers and analyzed with the given methodology. Finally the Result says that instead of using all the variables it will give good and reliability result by Naive Principal Components Analysis. This paper recommends that if there are more number of variables in a study Naive Principal Component Analysis perform well with minimum amount of loss of Information.

Acknowledgment

The second author acknowledges University Grants Commission, New Delhi for providing financial support to carry out this work under the scheme of Basic Scientific Research Fellowship.

REFERENCES

[1] Blischke, W.R., and D. N. P. Murthy, (2000). Reliability: Modeling, Prediction, and Optimization. Wiley Series in Probability and Statistics.

[2] Goel, A.L. (1980). A Software Error Detection Model with Applications. Journal of System and Software, 1, 243-249.

[3] Goel, A.L. (1985). Software Reliability Models: Assumptions, Limitations and Applicability. IEEE Transactions of Software Engineering, 12, 1411-1423.

[4] Jelinski, Z., and P. Moranda, (1972). Software Reliability Research", In Statistical Computer Performance Evaluation. W. Freiberger, Ed. New York: Academic, 465-484.

[5] Loganathan A and R. J. Muthuraj (2016).A new methodology for data reduction in software reliability studies. Communications in Statistics: Case Studies, Data Analysis and Applications, 2, 101-105.

[6] Loganathan A and R. J. Muthuraj (2017). Importance of Environmental Factors Affecting Software Reliability. Global and Stochastic Analysis, 4, 119-125.

[7] Patwa, S., and K. Malviya, (2014). A Survey on Factors Affecting Testing Techniques in Object Oriented Software. International Journal of Applied research on Information Technology and Computing, 5, 78-85.

[8] Pham, H. (2000). System Software Reliability. Springer Series in Reliability Engineering.

[9] Zhang, X., and H. Pham, (2000). An Analysis of Factors Affecting Software Reliability. Journal of Systems and Software, 50, 43-56.

[10] Zhu, M.Y., Zhang, X., and H. Pham, (2015). A Comparison Analysis of Environmental Factors Affecting Software Reliability. Journal of Systems and Software, 109, 150-160.