Fall 12-18-2014

Masquerade detection using Singular Value Decomposition

Sweta Vikram Shah
San Jose State University

Masquerade detection using Singular Value Decomposition

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Sweta Shah

The Designated Project Committee Approves the Project Titled

Masquerade detection using Singular Value Decomposition

by

Sweta Vikram Shah

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

Dr. Sami khuri	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Mr. Kunjan Kapadia	NetApp Inc.

Acknowledgements

I am very thankful to my advisor Dr. Sami Khuri for his continuous guidance and support throughout this project. Also I would like to thank the committee members Dr. Thomas Austin and Mr. Kunjan Kapadia for monitoring the progress of the project and their valuable time.

Abstract

Information systems and networks are highly susceptible to attacks in the form of intrusions. One such attack is by the masqueraders who impersonate legitimate users. Masqueraders can be detected in anomaly based intrusion detection by identifying the abnormalities in user behavior. This user behavior is logged in log files of different types. In our research we use the score based technique of Singular Value Decomposition to address the problem of masquerade detection on a unix based system. We have data collected in the form of sequential unix commands ran by 50 users. SVD is a linear algebraic technique, which has been previously used for applications like facial recognition. We present experimental results and we analyze the effectiveness and efficiency of this SVD-based masquerade detection.

Table of Contents:

List of Figures

List of Tables

Chapter 1

Introduction

Computer-system logs provide a glimpse into the states of a running system. Instrumentation occasionally generates short messages that are collected in a system-specific log. The content and format of logs can vary widely within different systems and also with components within a system [1]. Log Analysis is done to achieve goals like debugging, performance monitoring, fault detection, profiling and prediction. Log analysis can be used to find problems, define operational profiles, and even pro-actively prevent issues. Modern software-based systems collect information about their activity in logs. The information in the logs can therefore be used to trace the authenticity of the system's activities. Logs can be in any form like single files, collections of files, databases, or streams of data in a raw text format [2].

A strong logging infrastructure is essential for supporting the variety of applications is described here. It requires at least two features: log generation and log storage. Most general-purpose logs are raw or in an unstructured text format. The dataset used for this research is from one of the statistical science publications [4] by Professor Matthias Schonlau on intrusion detection in computer systems. This dataset provides UNIX command line data of 50 users collected over a period of several months. The dataset has been seeded with masqueraders, which have been used for the testing phase of our model.

The base for this research is to create a user profile indicating a genuine behavioral pattern and then test the actual behavior against the legitimate pattern to detect the abnormal behavior. The technique we use is Singular Value Decomposition (SVD), which is a matrix factorization technique. This technique takes a highly variable and diverse input matrix and reduces it to a smaller dimension, which will yet very well expose the substructure of the original input matrix [9]. The SVD is trained by a set of legitimate

commands provided as input in the form of a matrix, which gets transformed to a resultant matrix. This trained model is then used over the test data, which is a combination of both, legitimate and masquerade activities. We have done our experiments on the Schonlau's masquerade detection dataset for 50 users and got good results.

Further, chapter 2 will highlight the extensive literature survey that was done to support the research done. Chapter 3 talks about masquerade detection, the various techniques that can be used to carry out detection and its relation to machine learning. Chapter 4 explains the technique of Singular Value decomposition used for masquerade detection, the dataset and the algorithm. Chapter 5 and 6 contain the experimental results, its explanation and conclusion.

Chapter 2

Masquerade Detection

2.1 Introduction

In the modern world of computer security, a lot of effort has been put to prevent intrusions or attacks to a computer system. Masqueraders often attack systems that solely depend on password validations or login credentials. Once they have access to a user account they perform malicious activities, which they are not authorized for.

A masquerader is someone who tries to act like a legitimate user and secretly perform operations, which are unlikely to be performed by a normal, legitimate user. These attacks are often seen happening within an organization and they remain hidden and are difficult to find.

Intrusion detection performs vulnerability assessment to manage the security of computer networks and systems. It gathers information from several sources that record and maintain the system behavior on a timely basis and contain impressions of unusual or odd incidents.

In fig.1, A Network intrusion detection system is used to detect malicious activities by trapping traffic at crucial points of a web application that travels in and out of the system. This information is then analyzed to detect intruders in a system. These systems are easy to deploy as they do not change or affect the overall infrastructure of the system. One of the drawbacks is that a large amount of data is gathered over a period of time, which can get hard to manage.

Intrusion detection functions include:

- Monitoring and analyzing both user and system activities
- Analyzing system configurations and vulnerabilities
- Assessing system and file integrity
- Ability to recognize patterns that are typical of attacks
- Analysis of abnormal activity patterns in system
- Tracking user policy violations [7]

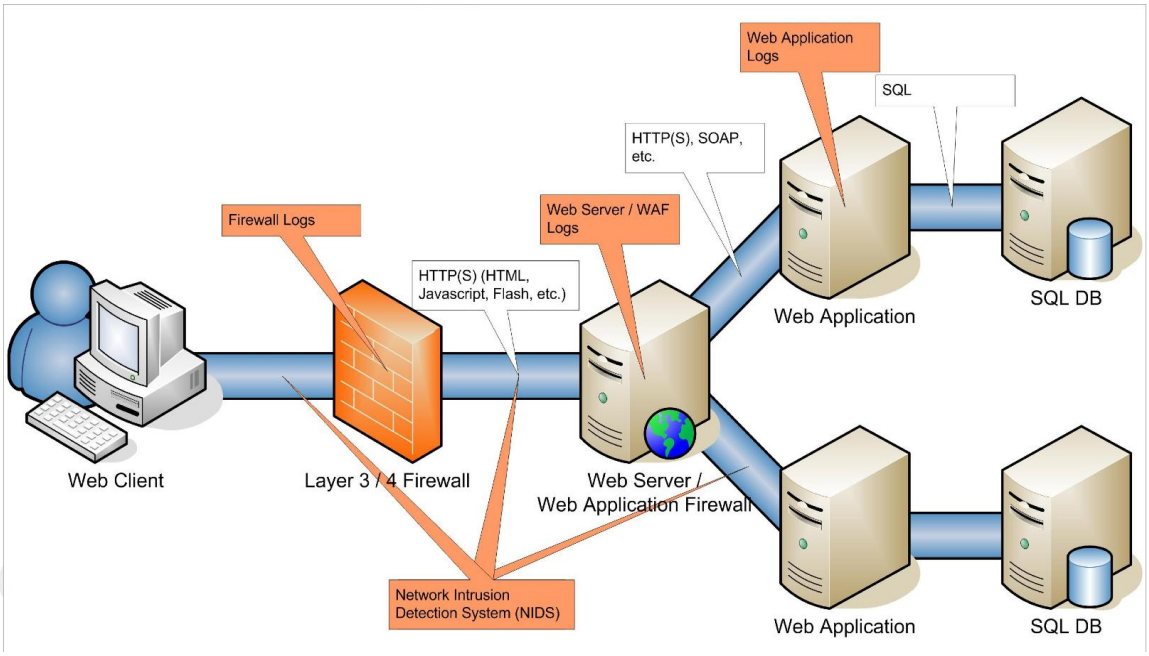


Figure 1. Detecting attacks on a network. [8]

There are two techniques of intrusion detection namely anomaly based and signature based intrusion detection.

- **Anomaly based IDS**: In an anomaly-based approach, the user profile or network traffic of the system is compared to a baseline formed by prior analysis carried out by the system. These baselines represent normal or acceptable states in a system. A massive or considerable deviation from this baseline generates an alarm in the intrusion detection system.

- **Signature based IDS**: In this technique, elements like user activity or network activities are compared against a database of signatures, which represent malicious activities recorded previously over the system. In case of a new attack, there will be a time gap for that attack to be added to the signature database.

In this research, we consider the anomaly-based technique of intrusion detection by maintaining user profiles with a baseline of valid user activity to which the actual data is tested against. The SVD helps in detection of variance between the baseline and actual data in the UNIX environment. We assume each user to have his, own peculiar style or behavior in running UNIX commands.

2.2 Machine Learning techniques:

Machine Learning techniques are widely used in various applications like artificial intelligence, bio-informatics, data mining etc. A machine gets trained from previous experiences and then provides prediction or feedback on new situations that are unknown. There are two learning approaches in machine learning.

1. Supervised learning:

   In our approach, SVD technique falls under the bracket of supervised learning where there is enough data available in advance that can be used to train a model for a good user. Once the model has extracted enough information from the existing dataset, it can be used to classify the valid user from the bad user (masquerader) in test data.

2. Unsupervised learning:

   In unsupervised learning, we do not train a model in advance with available datasets. It performs classification of data into clusters and those clusters are later assigned class names. It's an advantage in unsupervised learning, which can be used when you do not have the data available in advance or are unaware of the data fields. The results of unsupervised learning sometimes deviate from the human perception of a particular field or cluster to be important. It might end up considering other classes as important which are not expected.

2.3 Singular Value Decomposition

Singular Value Decomposition is a supervised machine learning technique, which has a very popular application in facial recognition. Facial recognition is an important field of study used for identifying criminals, digital image processing and so on. For facial recognition, the face can be decomposed into small feature images known as the eigenfaces. The space covered by these eigenfaces is known as facespace. SVD is a technique, which mainly identifies and orders the dimensions along those datapoints, which exhibit most variation [9]. On identifying the most variations, it is easily possible to generate a very good approximation of the original dataset. That is one of the reason SVD is used for data reduction.

Chapter 3

Literature Review

Intrusion detection aims at the timely discovery of any activity that jeopardizes the integrity, availability, or the confidentiality of an IT system. Based on the activity it observes, an intrusion detection system (IDS) may be either of three types: host, network, or application [3]. A host IDS is usually set to audit the functionality of the underlying operating system, as it executes system calls, but can also be set to watch critical resources. Based on the classification scheme, an IDS is either misuse or anomaly. According to author Garcia and K.A. Scotiabank the interest is in a problem, which is closely related to intrusion detection. The problem, which we call postmortem intrusion detection, is defined as follows: given a log file, find the point where the exploitation happened, if any. Thus, while intrusion detection tries to detect intrusions as they happen, postmortem intrusion detection monitors the system information that is offline with an intention to detect and locate the execution of the attack [3].

The postmortem IDS stated by author Garcia and K.A. Scotiabank use a novel combination of k-means and hidden markov model, which Garcia et altogether calls the KHMM. Since the IDS is a host based IDS, it also talks about T-stide. T-stide defines normal behavior by means of short-range correlations in the execution of system calls of a process. It is able to detect several common intrusions involving UNIX processes. T-stide uses a sliding window protocol to explore the user activity on the system, looking for any deviations from the profile of ordinary or expected system behavior [3].

The Garcia et al then suggests an approach to build an anomaly, learning-based, host-based intrusion detection model, where the working hypothesis is that an attack takes the form of an unusual sequence of system calls. In the test setup, the author assumes that the system has been compromised by an attack and that this intrusion has left some

information somewhere [3]. The first step of the mechanism is reduction where a lot of repetitive information is removed. Then, the approach uses a sliding window, which would take a certain threshold of events from beginning each time, and feed it to a classification system. The results of the classification system are compared with other techniques for performance checks. The classification is carried out using the KHMM technique [3].

The use of event co-occurrence scheme to preprocess the data before performing principal component analysis is one of the ways of intrusion detection. There are two steps to decide an input data as normal or intrusive. The initial step is extraction of features of the data. The second step is classification of the extracted features of the data. ECM (Event Co-Occurrence Matrix) represents the causal relationship by converting a section of an event sequence to an event co-occurrence matrix. An element of the co-occurrence matrix represents the occurrence of pair of events in the event sequence within a scope size [5].

On feeding the ECM, the principal component analysis technique will generate an eigen co-occurrence matrix which is the output of the learning phase. On projection of the event co-occurrence matrix on the Eigen matrix, Y.Oyama obtains the feature vector. The main feature of their implementation was the co-occurrence matrix representation of the feature and the reduction using PCA [5].

Log Analysis is crucial and fundamental to System Administration in today's world. Logs provide with an insight on exactly what is happening with the system that is being monitored. The article talks a lot about how as the systems vary the instrumentation that writes the log also varies. It discusses the advancements and the challenges associated to log analysis [1].

The paper provides an idea of some of the most common applications of log analysis, describes some of the log types which can be analyzed and the methods of analyzing them,

and describes some of the challenges associated with it [1]. First, is debugging, in which he talks about some previous techniques like “printf” or “grep” commands that were used to understand the flaws of the system. The simplest and most common use for a debug log is to grep for a specific message. If a server operator assumes that a program crashed due to a kind of a network failure, then he would try to look for a message in the server logs [1]. But debugging gets difficult when we don’t know what to search for or the log messages individually are misleading. Second, is performance monitoring, in which log analysis can help us improve resource allocation, bottleneck situations and preparing resource usage statistics for long term review.

Security is another very useful application where logs can help detect fraudulent activities. Intrusion detection is the term that is used to describe this kind of log analysis. This approach needs real time monitoring of the data that will help prevent such mishaps. Log analysis for security may be signature based, in which the user tries to detect specific behaviors that are known to be malicious; or anomaly based, in which the user looks for shift from typical/normal or good behavior [1]. The other applications discussed are prediction and reporting and profiling. User’s footprint over the Internet is a very valuable piece of information and it can help detect shopping patterns, likes, and dislikes of the consumer. Logs that monitor characteristics of tasks from a cluster’s workload can be used for resource utilization profiling at large data centers that compute terabytes of data which cannot afford any downtime [1].

The paper also speaks about how the machine learning techniques can be used for the various applications dealing with log analysis. Machine-learning techniques, especially anomaly detection, are normally used to monitor failure messages that can help root cause issues in the system. Machine-learning tools usually require input data as numerical feature vectors. It is a cumbersome task to convert free-text log messages into meaningful feature vectors that can used to analysis of the system under concern [1].

Chapter 4

Singular Value Decomposition for Masquerade detection

4.1 Eigen Vectors and Eigen Values

An Eigen vector is a non zero value of a particular square matrix say A, such that the multiplication of that vector x with A is equal to a scalar value multiplication of the vector. The standard equation is as given below.

$$Ax = \lambda x$$

In Linear Algebra, if a vector satisfies the above equation, then x is considered to the eigen vector of *A* and λ as the eigen value associated to the vector *x*. The figure given below would explain the relation [19].

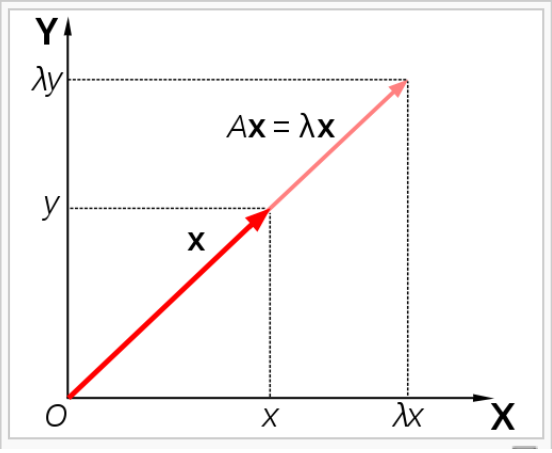


Figure 2: Eigen vector x and Eigen value λ

In the diagram above we see, that matrix *A* stretches vector x where the direction of x does not change but only the magnitude changes. Thus x is a vector of *A*. In our technique, the user commands are parsed and placed in a matrix A, and the variation of data in the matrix is placed in covariance matrix [19]. The eigen vectors and eigen values for these matrices is calculated and these eigen vectors are then projected into a space called as the eigenspace.

More the value of the eigen vector, more important would be its corresponding eigen vector in contributing to the variance of the dataset that we provided. Thus, SVD uses eigen vector and eigen values to understand the high variance points in the data.

4.2 Singular Value Decomposition for Face Recognition

In Face Recognition, the images are expressed in terms of a covariance matrix and the eigen vectors for these matrix are calculated. The eigen vectors which represent the variations in the images are projected on a space called the face space. These vectors enclose this face space. When an image is projected on to the face space, the original image can be reconstructed using the eigen vectors [17]. In facial recognition technique after the column vector is constructed with the pixels of individual image, we subtract the mean pixel value of all the training input images at that position from the actual pixel value of all the images which was being projected.

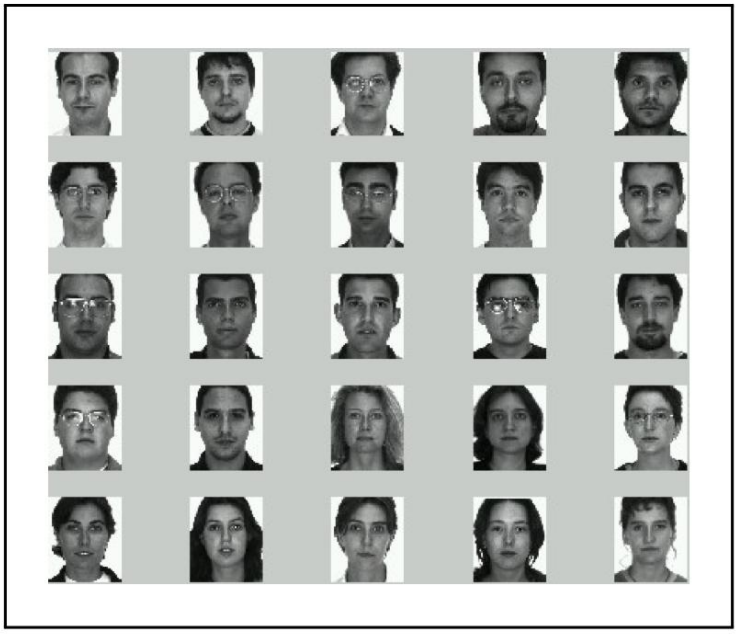


Figure 3: Training Images [10]

The next figure contains the eigen faces for the training input set of images shown above. These images are further used for projection.

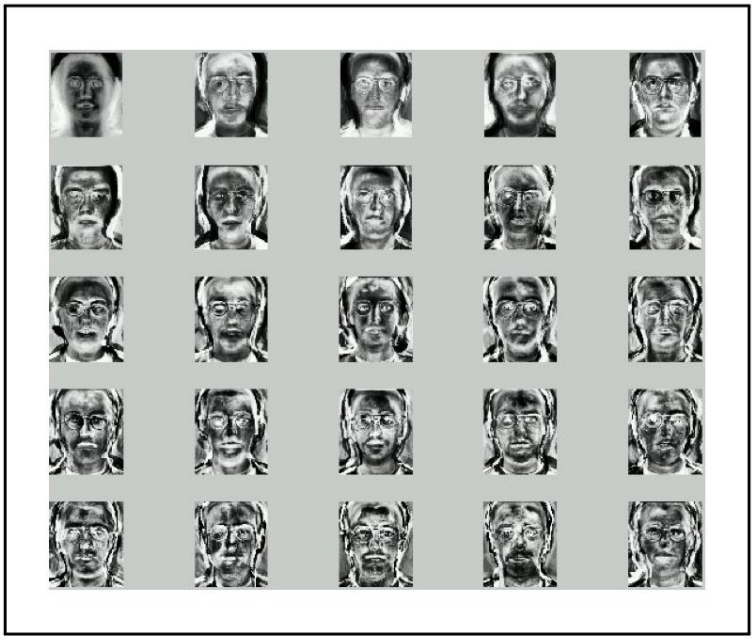


Figure 4: Eigen faces of the images [10]

We would now look at the projection of images. When a known, legitimate image is projected on the eigen space, the original image can be reconstructed. But when an unknown, invalid image is projected on the eigen space, the original image cannot be reconstructed. The next image shows the projection image for a known as well as an unknown image on the eigen space [17].

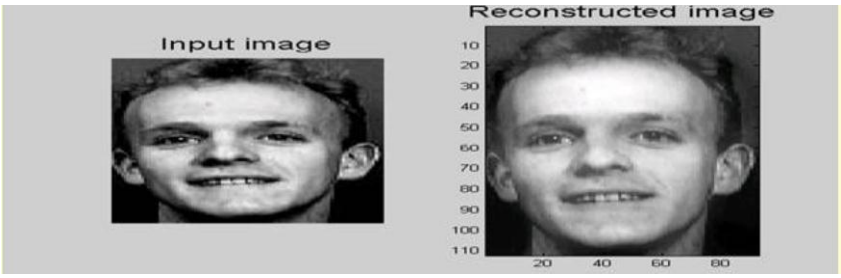


Figure 5: Projection of a known image

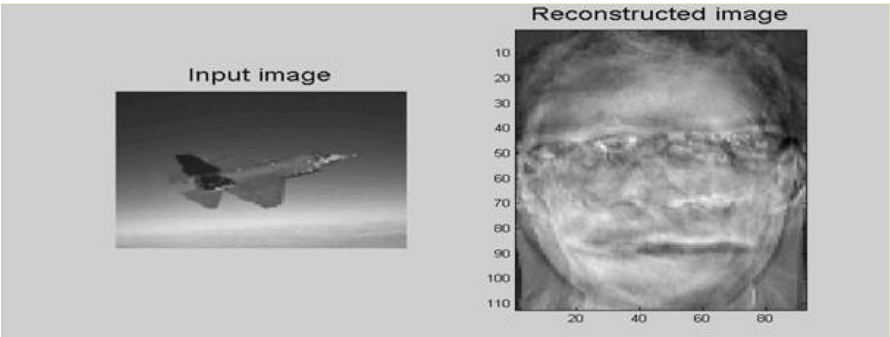


Figure 6: Projection of an unknown image

4.3 Schonlau's Dataset

The dataset consists of UNIX commands ran by 50 different users. For each user the dataset consists of 15k commands. Genuine users run the first 5k commands and the remaining 10k commands are seeded with masqueraders and our algorithm aims at identifying the masqueraders from those 10k commands. The 10k commands are thought of as 100 blocks of 100 commands each. The algorithm would try to locate the masqueraded block out of the 100 blocks of commands.

After the first 5000 commands, the probability of a masquerader block starts with 1%. If a block is a masquerader block, there is a probability of 80% that the next block too might be a masquerader block. Up to 5% of the test data i.e. the 10k commands can be masquerader blocks. The website also provides the location of masqueraders in the dataset to validate the results of the masquerade detection algorithm. The file has 50 columns for 50 different users, where each column has 100 values for 100 blocks of test data. The values in the columns are in the form of 1's and 0's. 1 represents a masquerade block and 0 represents a genuine user block.

Sample Dataset:

- Here is a sequence for legitimate user commands for User 3 of the dataset.

cpp
sh
xrdb
cpp
sh
xrdb
mkpts
env
csh
csh
csh
sh

- This is a sequence of commands by the masquerader who pretends to be User 3

egrep
egrep
egrep
java
java
make
.java_wr
expr
expr
dirname
basename
egrep
egrep
egrep
egrep

The difference in the commands as well the sequence in which they are run would highlight this block as a masquerade block in the results of our implementation.

- Example of SVD

SVD is based on linear algebraic expression which states that a rectangular matrix can be broken down into three matrices – a diagonal matrix U, a diagonal matrix S and a transpose of orthogonal matrix V.

$$A_{mn} = U_{mm} S_{mn} V_{nn}^{T}$$

Starting with a matrix A,

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

So, inorder to find U, we need transpose of A

$$A^{T} = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

$$AA^{T} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

Lets now find the eigen values and eigen vectors of the above matrix.

$$\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Forming an equation,

$$11x_1 + x_2 = \lambda x_1$$

$$x_1 + 11x_2 = \lambda x_2$$

Rearranging the equation gives us,

$$\begin{vmatrix} (11-\lambda) & 1 \\ 1 & (11-\lambda) \end{vmatrix} = 0$$

Finally we get,

$$(11-\lambda)(11-\lambda) - 1 \cdot 1 = 0$$
$$(\lambda - 10)(\lambda - 12) = 0$$
$$\lambda = 10, \lambda = 12$$

Now we would substitute the eigen values in the equation,

On substituting 12,

$$(11-12)x_1 + x_2 = 0$$
$$x_1 = x_2$$

On substituting 10,

$$(11-10)x_1 + x_2 = 0$$
$$x_1 = -x_2$$

Thus the final matrix is where column one is eigen vector for value 12 and the second column is for value 10:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

On applying orthonormalization we get U as,

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Similarly we need to find transpose of A x A for matrix V and follow the same steps. That gives us,

$$V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

The non-zero eigen values of U and V form the diagonal elements of the diagonal matrix S. These valus are same for both U and V. We also add a zero column vector for proper dimensions.

$$S = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

On multiplying all the three matrices we end up getting the original matrix A.

$$A_{mn} = U_{mm} S_{mn} V^T_{nn} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{\sqrt{12}}{\sqrt{2}} & \frac{\sqrt{10}}{\sqrt{2}} & 0 \\ \frac{\sqrt{12}}{\sqrt{2}} & \frac{-\sqrt{10}}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

This example clearly illustrates how a matrix gets decomposed into matrices [9] using the Singular Value Decomposition technique, which has been used for the user commands in the context of our project.

4.4 Algorithm for masquerade detection

According to the dataset, the first 5K commands are legitimate, out of which we use the first 2K commands for training the system and the remaining 3K commands to form a threshold of genuine data. SVD would be applied to the training set of 2K commands.

1. The training set gets divided into 20 blocks of 100 commands each forming an input matrix of 100 x 20. On applying SVD to this matrix, we get matrices for U, V and Sigma(S). Matrix A is given by Matrix U multiplied by Matrix S (Sigma) multiplied by transpose (T) of Matrix V. The size of each of them is 100 x 20, 20 x 20 and 20 x 20.

$A = U S V^T$

Where, A is 100 x 20

2. The matrix U contains vital information about the input matrix A. We need to project the input matrix A on an Eigen space which is U^T.

U^T x A = Output of training phase [20 x 20]

3. The Threshold data set of 3k commands are divided into 30 blocks of 100 commands each which is 100 x 30 in dimension. We now project the threshold dataset on the eigen space (U^T).

U^T x B = Projection of Threshold data on the Eigen space [20 x 30]

4. Euclidean distance will be calculated between the threshold data projection and the training output where every column of threshold set will be compared to every other column of the training output and the minimum value would be selected. This results in generating the distance between training data and threshold data in the form of 30 scores.

5. The test data set consisting of 10k commands are divided into 100 blocks of 100 commands each and are projected on the Eigen space just like the threshold dataset.

U^T x T = Projection of test data on the Eigen space [20 x 100]

6. Euclidean distance will be calculated between the test data projection and the training output where every column of test set will be compared to every other column of the training output and the minimum value would be selected. This results in generating the distance between training data and test data in the form of 100 scores.

7. The threshold dataset scores and the test dataset scores are then projected on a graph. The threshold scores form a threshold indicating the boundary, which represents commands from a legitimate user. The test data scores are projected to see instances where they are far apart from the threshold data scores, which indicate the masquerade blocks of data.

The scores from the test data were then compared to Schonlau's result validation file, which provides the location of all the masqueraders in the test dataset. This helped to verify the performance of the algorithm on the dataset. The experiments were run on all 50 users and results would be seen in the next section.

Chapter 5

Experiments

The charts contain the plotting of the threshold scores and the test data scores on a scatter plot. The main observation on the chart would be to see points where the test data scores are far apart form the threshold data scores.

Fig 6 shows results for evaluating masquerade out of set of legitimate commands. On running the technique, the results contain 30 values that indicate the threshold and 100 values that represent the test data scores. Series 2 represents a single value for the threshold, which is an average of threshold values. This result indicates that none of the test data scores surpass the threshold, which indicates there has not been a masquerade attack for User 1.

User 1

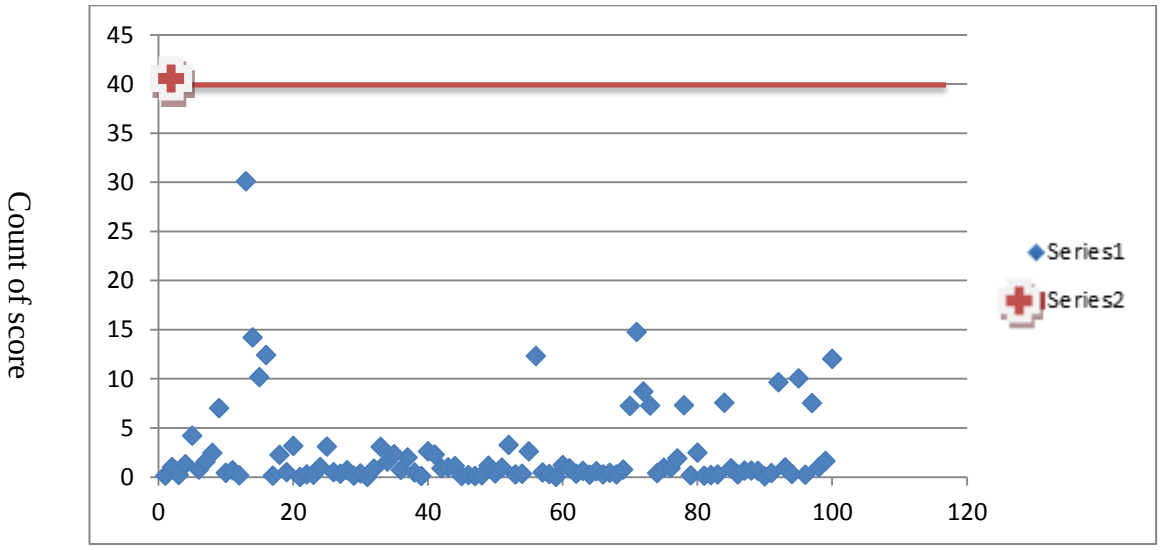


Figure 6: Scores for User 1

In fig 7 for User 2, the threshold value is 10.57 so all the test data points, which cross this range, would be considered to be as masquerade data points. Block 61 and 62 in the chart are far beyond the threshold and that is why these commands blocks of test data would be considered as masquerade blocks. The test dataset of 10k commands get divided into 100 blocks of 100k commands each. The result indicates that block 61 and 62 are blocks of commands ran by the masquerader.

User 2

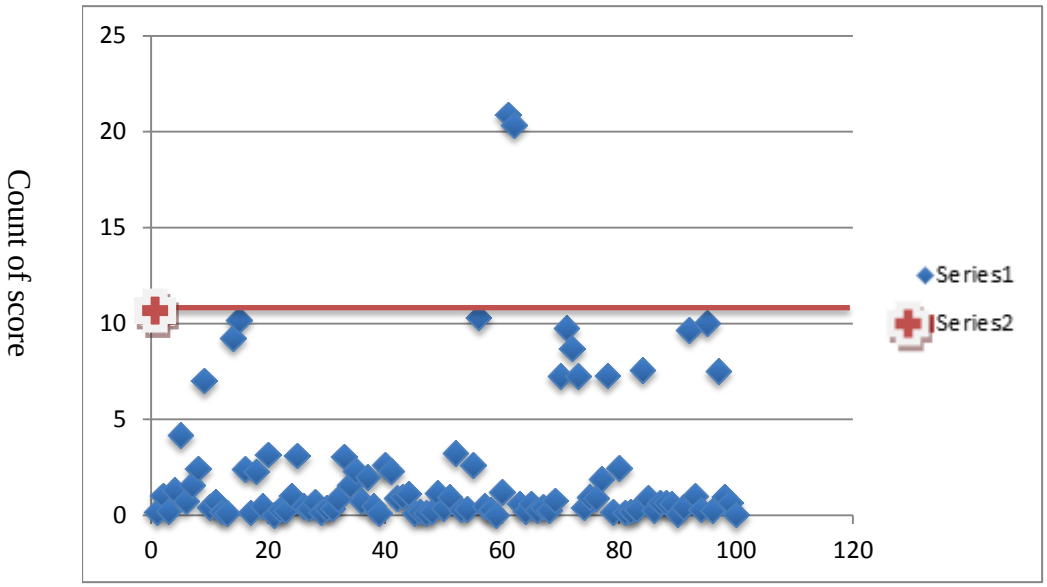


Figure 7: Scores for User 2

- 5000 blocks of commands with 100 commands in each block were tested as a part of the Testing Phase. Each of those blocks was identified as a masquerade or a non-masquerade block of commands. Some of the blocks were identified as false positive alarm or false negative alarm. Table 1 indicates the total number of blocks in each category and the users that exhibited those alarms.

- Analysis of results for 5000 blocks [False Positives, False Negatives/Error Free]:

Category	Number of blocks (Sum of blocks of users in a category)	Users
False Positives	13	6,37
False Negatives	754	2,7,10,15,16,24,25,26,34,36,42,45, 48
No False Positives/Negatives	2133	27,29,31,32,33,35,38,39,40,41,43, 46,49,50

Table 1: False Positive/Negative statistics

Table 1 Specifies results for users, which had more intruder alarms than expected (False positives), lesser alarms than expected (False negatives) and those, which had no false positives or negatives. Rest of the command blocks (4233 blocks), were accurate and were neither false positive/negative or may belong to the category of having no masquerades. These include all the blocks that were identified as a masquerade or a non-masquerade block in the complete testing phase and if that identification was valid.

- Statistics for masquerade free files:

	No.	Out of total users (%)	Error
Masquerade Free Files in the dataset	21	42%	No

Masquerade Free Files detected by the system	21	42%	

Table 2: Evaluation for Masquerade Free Files

Table 2 indicates that our implementation of the technique could identify all the users, which did not have any masqueraders attacking them exactly. Out of 50 users, there were 21 users with no masquerader commands.

- Statistics of error free results

	No. Of users	Out of total users(%)
Exact results (Results with no false or missing alarms)	35	70%

Table 3: Evaluation of Exact results

Table 3 shows there were exactly 35 users for which the results had no false positives and negatives. Based on the data in table 1 and table 2 exact results is the combined sum of masquerade Free Files and no False Positives/Negatives data set. Table 3 also shows 70% of total data is exact determination of the results. They are identified exact masquerader blocks as per the result set on Schonlau's page [11].

- System Setup for execution:

The code implementation for the SVD technique is done in Java using the Java SE development Kit 7. JAMA library was used for all the matrix computations involved in the technique. The code has been divided into two modules, the Trainining Model Phase and the TestExecution phase. The training model processes the training dataset and the test execution is done on the threshold dataset and the test dataset. The system configuration is as follows.

Operating System: MacOS X version 10.9.3

Processor: 2.6GHz Intel Core i7

Memory: 16GB

Program runtime for 1 User: 2101 milliseconds (35.01 seconds)

Chapter 6

Comparison other techniques

The statistical science research paper implemented by Prof. Schonlau [4] and other scientists has compared several other techniques on the same dataset. There is also a well-defined comparison of these different statistical approaches. Each of these techniques have their own thresholds on the basis on which the false positives and negatives get calculated. The techniques are as follows:

1. Uniqueness
2. Bayes one-step Markov
3. Hybrid multistep Markov
4. Compression
5. Sequence match
6. IPAM

The best performing techniques were the Bayes one-step and Hybrid multistep technique.

Bayes one-step Markov:

It is based on one-step transitions from one command to the next. It uses a Bayes factor statistic to test the null hypothesis that the observed one-step command transition probabilities [20].

Hybrid multistep Markov:

It is a combination of markov chain and the independence model. Whenever Markov models become unusable on the dataset, in such cases a model of probabilities is used.

Uniqueness: The uniqueness approach is based on the idea that commands not previously seen in the training. Data may indicate an attempted masquerade. Moreover, the fewer

users that are known to use that command, the more indicative that command is of a masquerade [20].

Compression:

This technique creates a mapping of training set and the testing set to the amount of compression possible. The differentiating factor would be the fact that data different form the restricted set of training set would not compress as well as the training set [20].

IPAM (Incremental probabilistic action modeling):

This technique depends on one step command change probabilities estimated using the training dataset. This model keeps updating itself with modifications to the training step updating the overall analysis [20].

Sequence Match:

In this technique, a similarity measure is computed between the recent commands and the existing user's profile. The user's profile too is a sequence of 10 commands used by the user in the past. The similarity measure is a match comparison between the user profile and the new set of commands [20].

Statistics of each technique:

Method	False Alarms(%)	Missing Alarms(%)
Uniqueness	1.4	60.6
Bayes one-step Markov	6.7	30.7
Hybrid multistep Markov	3.2	50.7
Compression	5.0	65.8
Sequence-Match	3.7	63.2
IPAM	2.7	58.9
SVD	**4.0**	**26**

Table 4: Performance statistics of other technique [20]

Table 4 above shows the performance of other intrusion detection techniques compared to Singular value detection (SVD). As per the table 4 above, 4% is false alarms and 26% for the missing alarms for SVD. This technique looks strong when looked at the data on missing alarms. The highlighted technique (SVD) in the table comes from the experiment performed in this project.

As per their research, Hybrid multistep did the best followed by Bayes one step Markov and then comes Uniqueness. One of their observations in the results was that as the false alarm rate went down, the missing alarm rate went up. SVD thus proves to be better than Bayes one-step Markov, but poorer to Compression as far as false alarms are considered. In case of missing alarms, it is better than all the other techniques but slightly less efficient than Bayes one-step markov [20].

ROC curves:

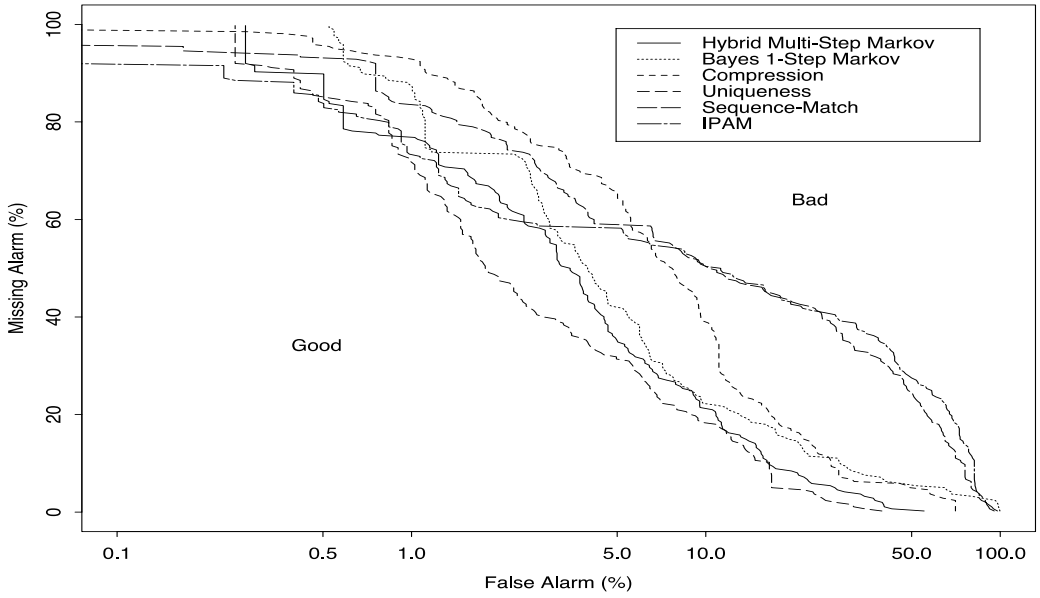


Figure 57: ROC for other technique [20]

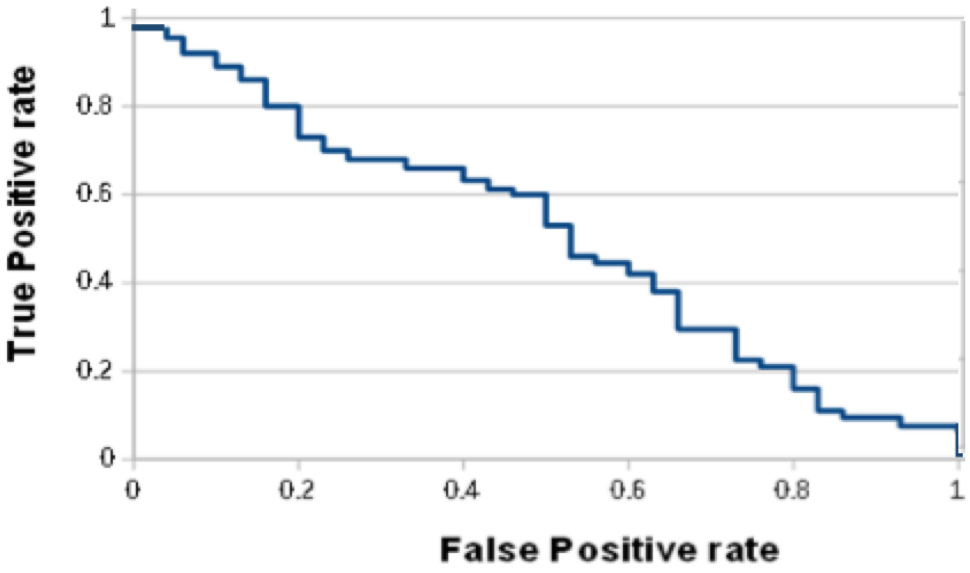


Figure 58: ROC for SVD

Chapter 7

Conclusions and Future Work

Other machine learning techniques have been used in the past to carry out masquerade detection on user data. Our technique helps at user profile building, where per user analysis helps us to identify masqueraders for that particular user. This approach can help in the protection of a huge system where there are different users using a system. The admin can use the application to identify intrusions for a specific user at specific time instances. Our implementation works well on large datasets with huge variety in the dataset. It identifies crucial data points to create a small yet effective representation of the original dataset. We have done experiments on 50 different user's dataset to validate the technique's performance. The technique works fast and efficiently.

One of the future improvements that can be done to this technique is adding the factor of conceptual drift to this technique. Conceptual drift is a concern in every machine learning technique, where every change in user behavior should not be considered as masquerade intrusion. Optimizations can be made to adapt to this change in user behavior, where there can be a distinguishing factor between user behavioral change and intruder actions. One of the approaches to achieve this can be done by updating the training model frequently using a certain kind of feedback mechanism on regular intervals. This help the system adapt to new changes in the user profile.

Another improvement that could be done is real time dashboards that run the application on the log files in real time. Whenever there is a spike in the commands being tested on the SVD machine, alerts can be generated to warn the admin of the centralized or shared system.

List of References

[1] Adam Oliner, Archana Ganapathi, and Wei Xu in 2012. Advances and challenges in log analysis.*Commun. ACM* 55, 2 (February 2012), 55-61. DOI=10.1145/2076450.2076466 http://doi.acm.org/10.1145/2076450.2076466

[2] Meiyappan Nagappan. 2010. Analysis of execution log files. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2* (ICSE '10), Vol. 2. ACM, New York, NY, USA, 409-412. DOI=10.1145/1810295.1810405 http://doi.acm.org/10.1145/1810295.1810405

[3] Garcia, K.A. Scotiabank, R Monroy . Trejo, C Mex-Perera in Nov 2012 analyzes Log Files for Postmortem Intrusion Detection, Systems, Man, and Cybernetics in Part C: Applications and Reviews, IEEE Transactions

[4] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi in 2001 presented a paper on computer intrusion for Detecting masquerades.

[5] M. Oka, Gar, and K. Kato in 2004 presents on Eigen co-occurrence matrix method for masquerade detection. In Publications of the *Japan Society for SoftwareScience and Technology*

[6] Large Scale click stream and transaction log mining in practice in October 6-9,2013. http://cci.drexel.edu/bigdata/bigdata2013/IEEE.BigData.Tutorial.2.slides.pdf

[7] Intrusion Detection, SearchMidmarketSecurity in May 2007. http://searchmidmarketsecurity.techtarget.com/definition/intrusion-detection

[8] Meyer Roger in 26 January 2008 detected attacks on web applications from log files, *Sans Institute InfoSec Reading Room*

[9] Kirk Baker in January 2013 illustrated Singular Value Decomposition Tutorial by http://www.ling.ohiostate.edu/~kbaker/pub/Singular_Value_Decomposition_Tutorial.pdf

[10] Original Training images and their Eigen faces by Santiago Serrano at Drexel University http://www.pages.drexel.edu/~sis26/Eigenface%20Tutorial.htm

[11] Matthias Schonlau dated 2003 presented a file for Masquerading User data http://www.schonlau.net/

[12] Weisstein, W Eric in October 2014 presented Singular Value Decomposition from MathWorld—AwolframWebResource. http://mathworld.wolfram.com/SingularValueDecomposition.html

[13] JAMA, Java Matrix Package live paper latest updated November 23, 2012
http://math.nist.gov/javanumerics/jama/

[14] Tutorial on Eigen values, Eigen vectors and Eigen spaces at Georgia Tech University
http://people.math.gatech.edu/~xchen/teach/lin_alg/Eigen.pdf

[15] Zhanchun Li, Zhitang Li, Bin Liu in Nov 2006 wrote a paper on Masquerade Detection System Based on Correlation Eigen Matrix and Support Vector Machine on *Computational Intelligence and Security, 2006*

[16] Han-Ching Wu, Huang, S.-H.S in Nov 2008 presented on User Behavior Analysis in Masquerade Detection Using Principal Component Analysis, *Intelligent Systems Design and Applications, 2008. ISDA '08.*

[17] Jidigam, Ranjith Kumar, Metamorphic Detection Using Singular Value Decomposition (2013).

[18] Rawat Sanjay, Pujari Arun.K, Gulati V.P. in Jan 2006 wrote on the use of Singular value decomposition for Fast Intrusion detection system, Science Direct, *Electronic Notes in Theoretical Computer Science 142* (2006) 215–228, 3 January 2006,

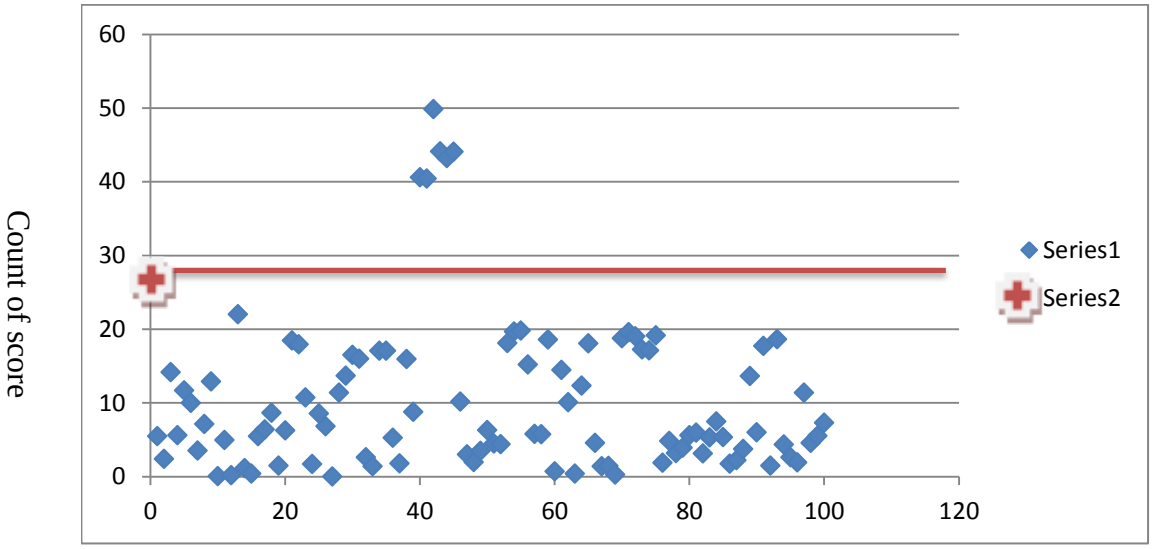
[19] Golub, Gene F, van der Vorst, Henk A in 2000 presented on Eigenvalue computation in the 20th century, *Journal of Computational and Applied Mathematics*

[20] Schonlau, Matthias, DuMouchel, Ju William , Wen-Hua Karr, F Alan , Martin Theusan, Vardi, Yehuda in 2001 presented on Computer Intrusion and Detecting Masquerades.
http://projecteuclid.org/euclid.ss/998929476.

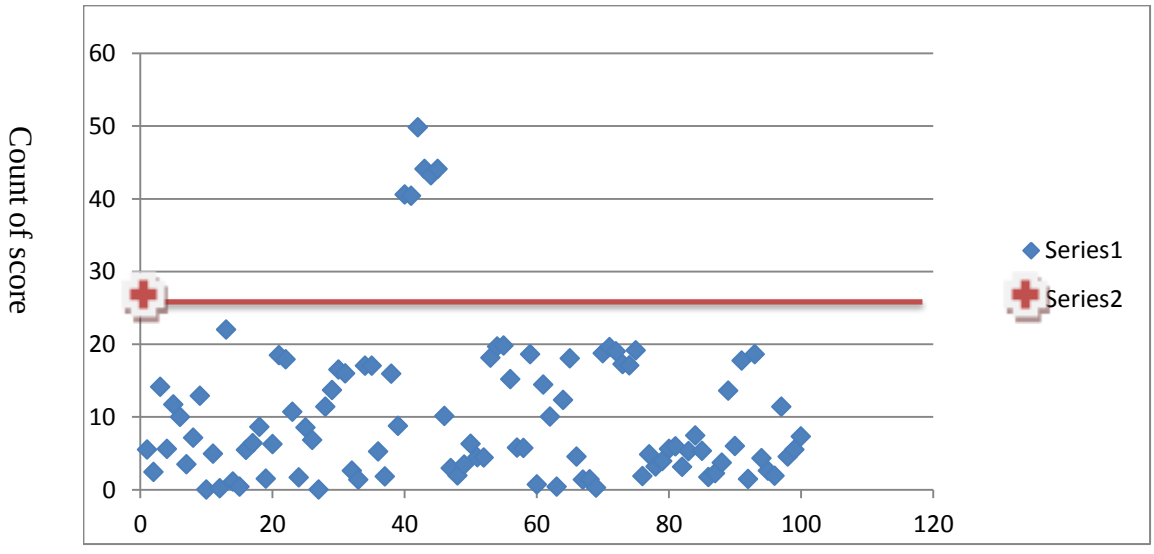
Appendix

Results from User 3 to User 50

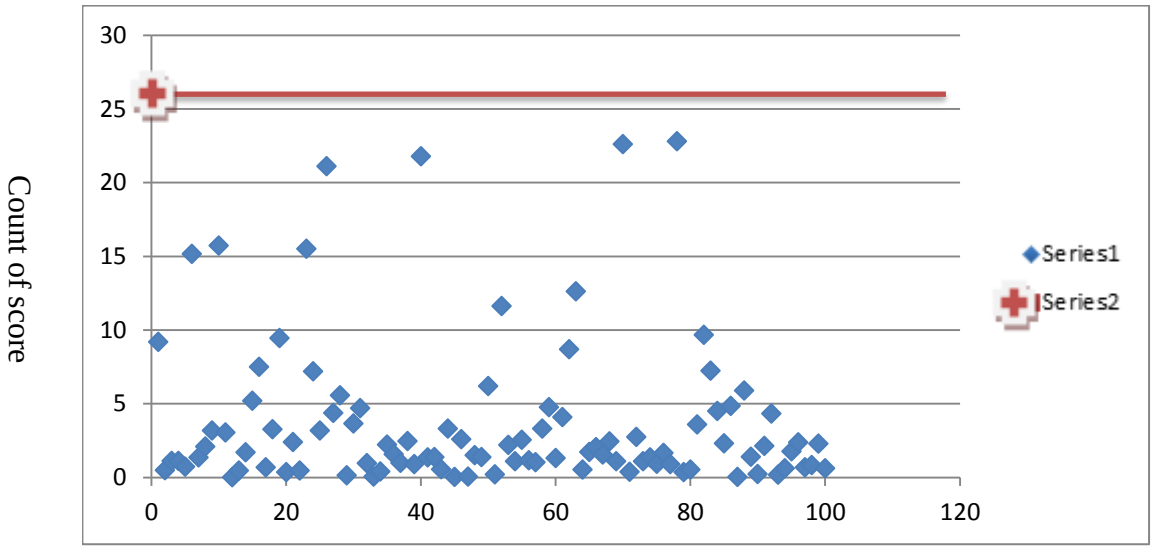
User 3



User 4

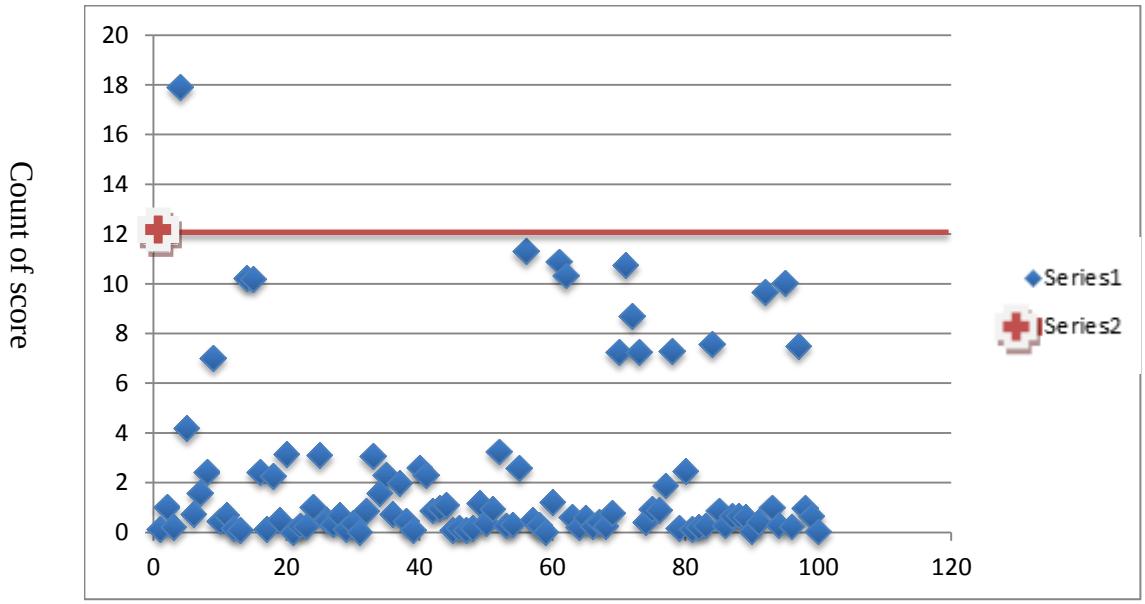


User 5



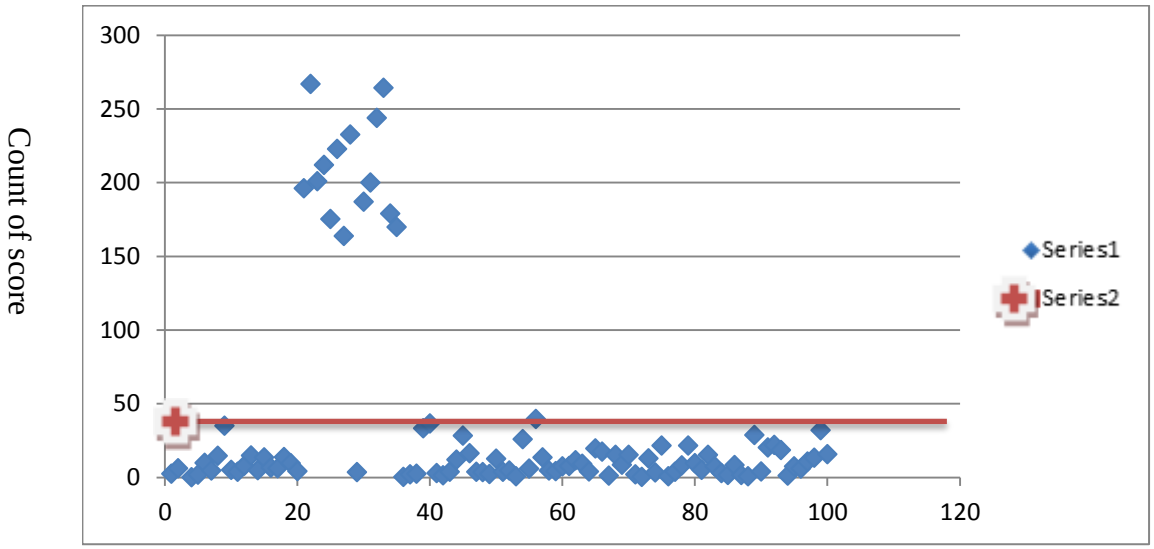
No. of scores

User 6

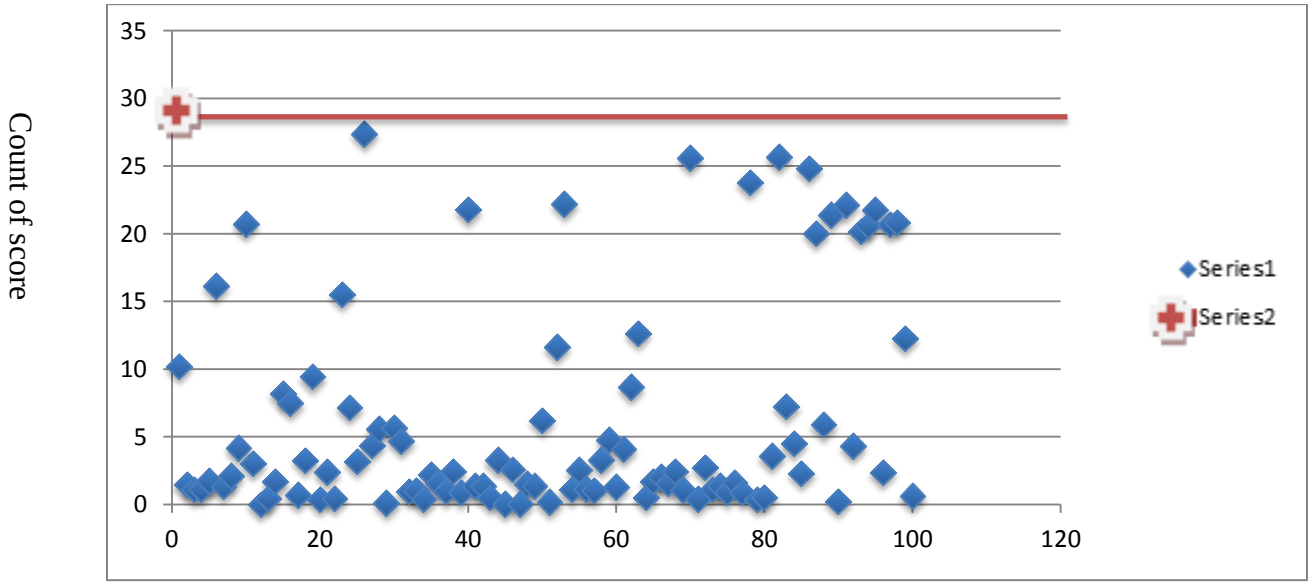


No. of scores

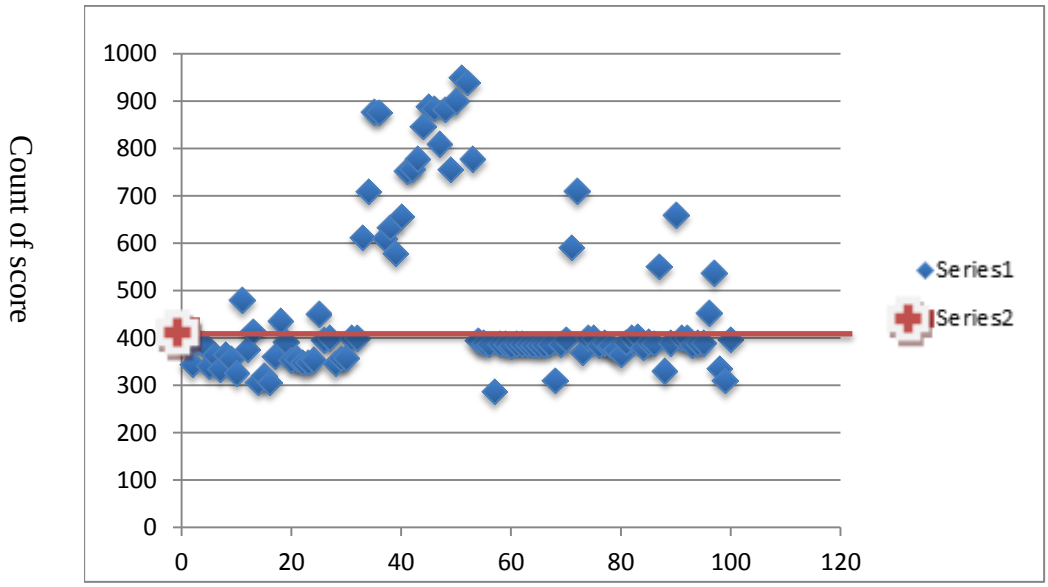
User 7


Count of score
300
250
200
150
100
50
0
0
20
40
60
80
100
120
Series1
Series2
No. of scores

User 8

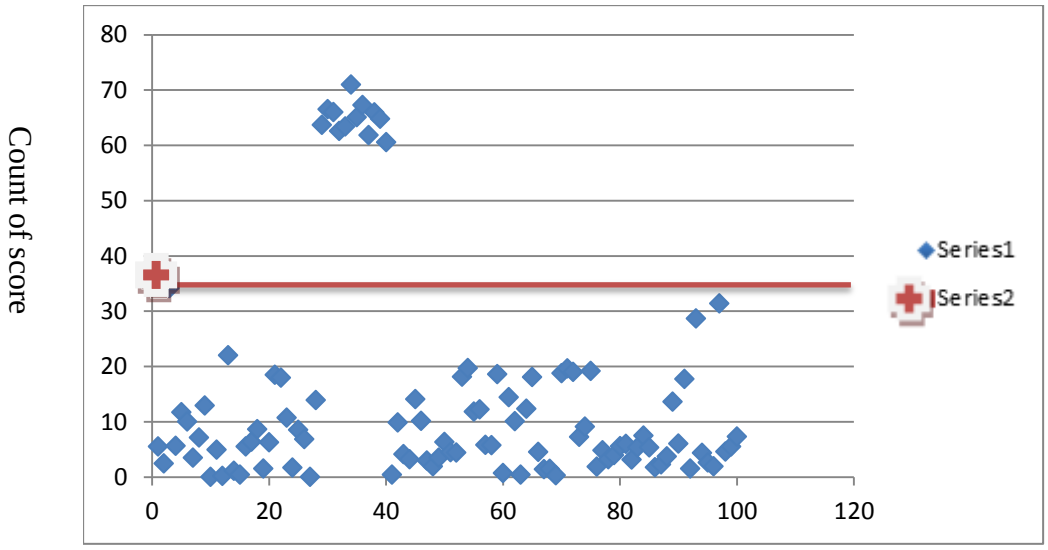

Count of score
35
30
25
20
15
10
5
0
0
20
40
60
80
100
120
Series1
Series2
No. of scores

User 9



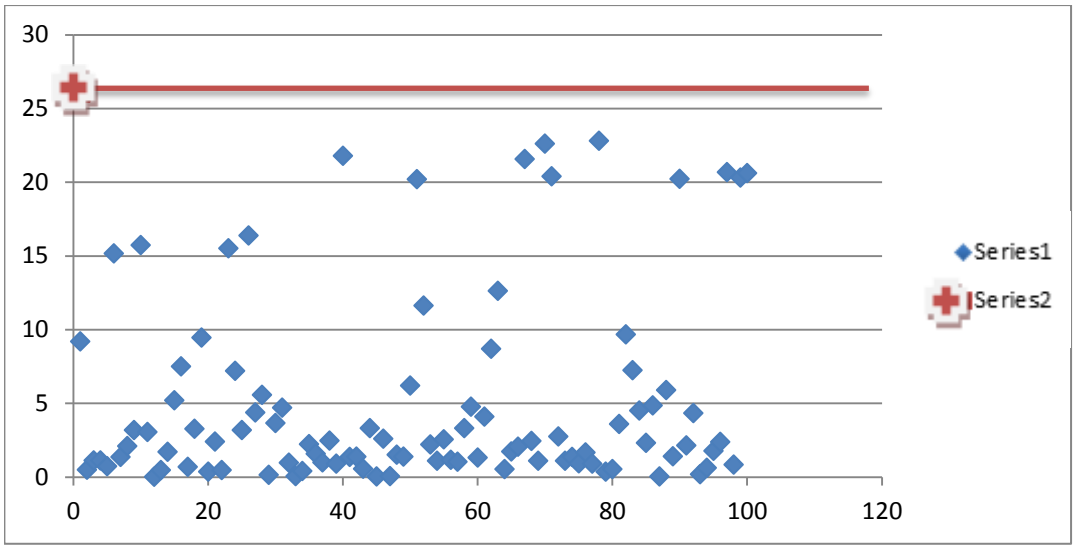
No. of scores

User 10



No. of scores

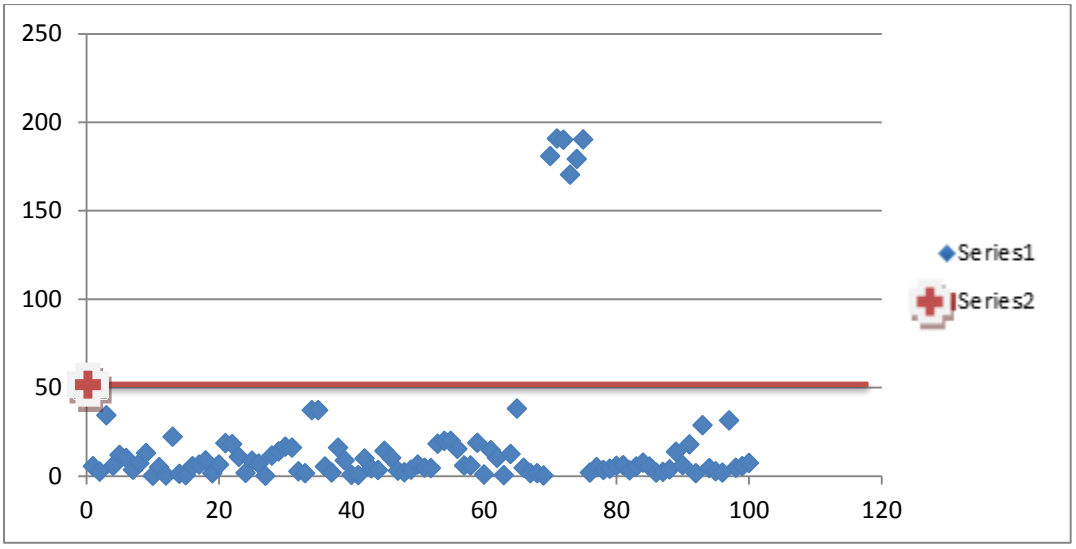
User 11



Count of score

No. of scores

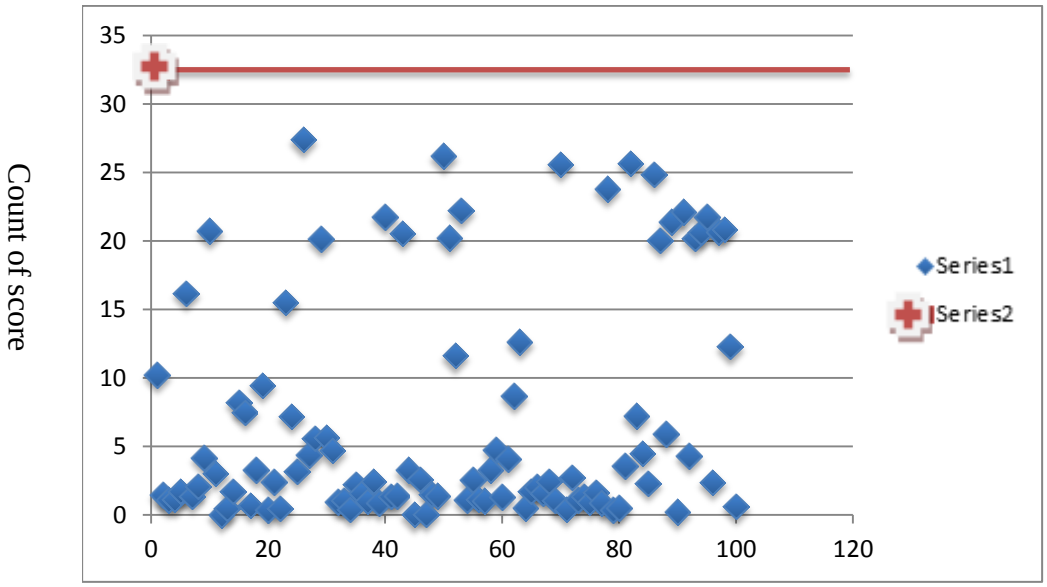
User 12



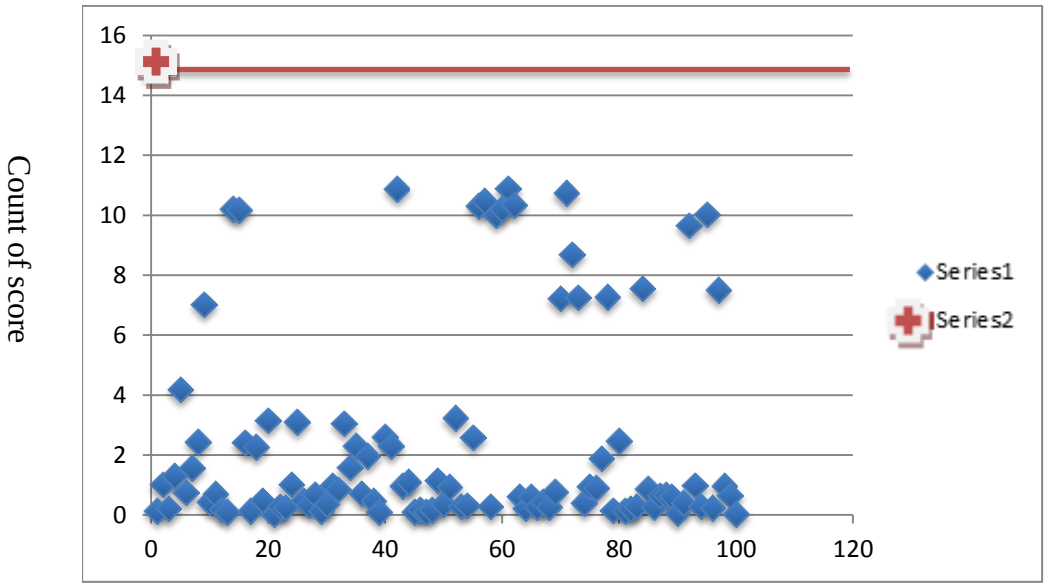
Count of score

No. of scores

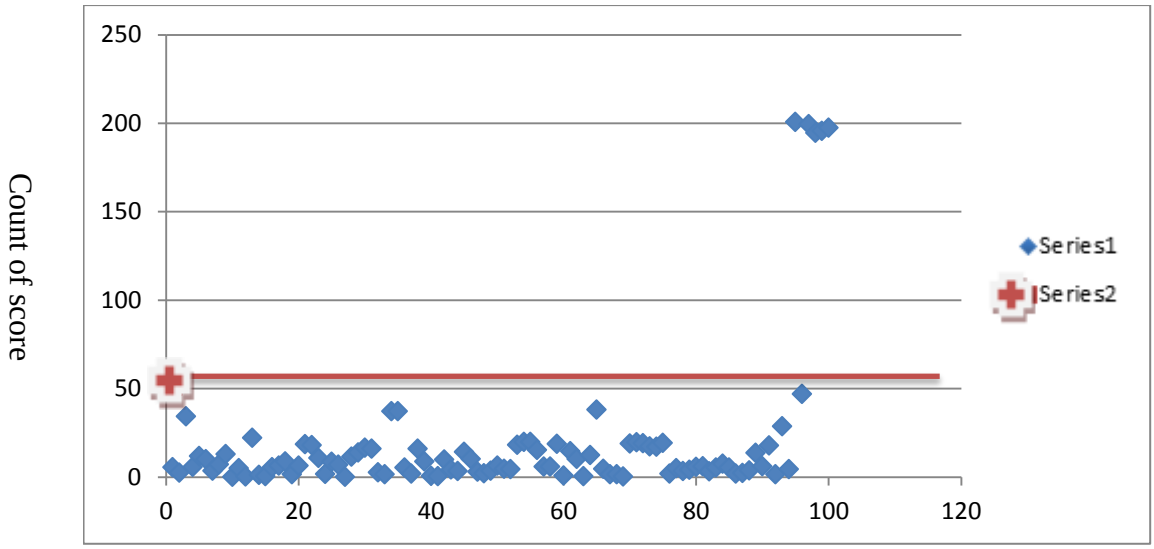
User 13



User 14

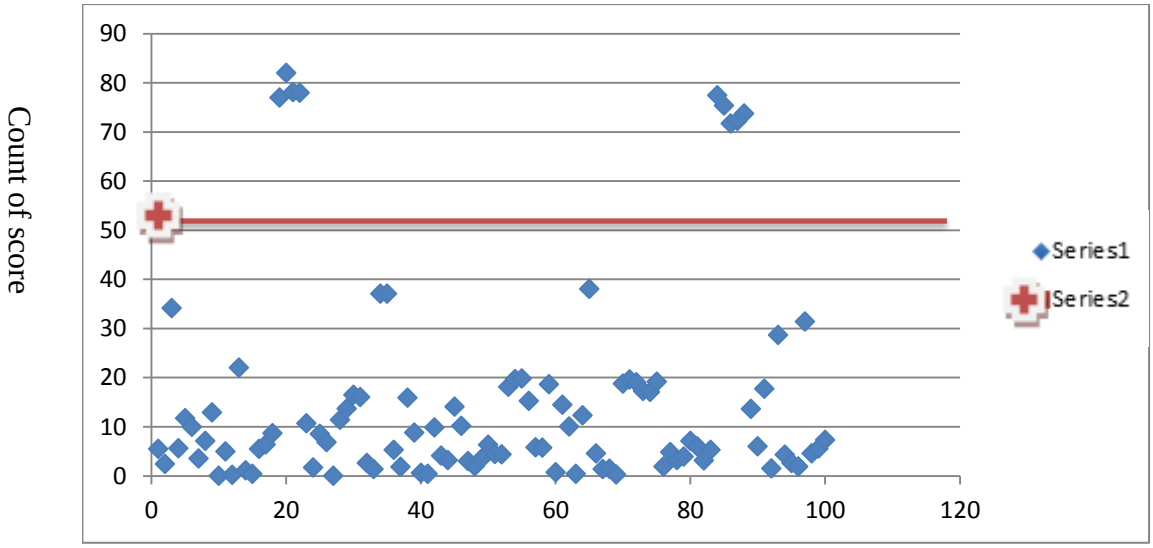


User 15



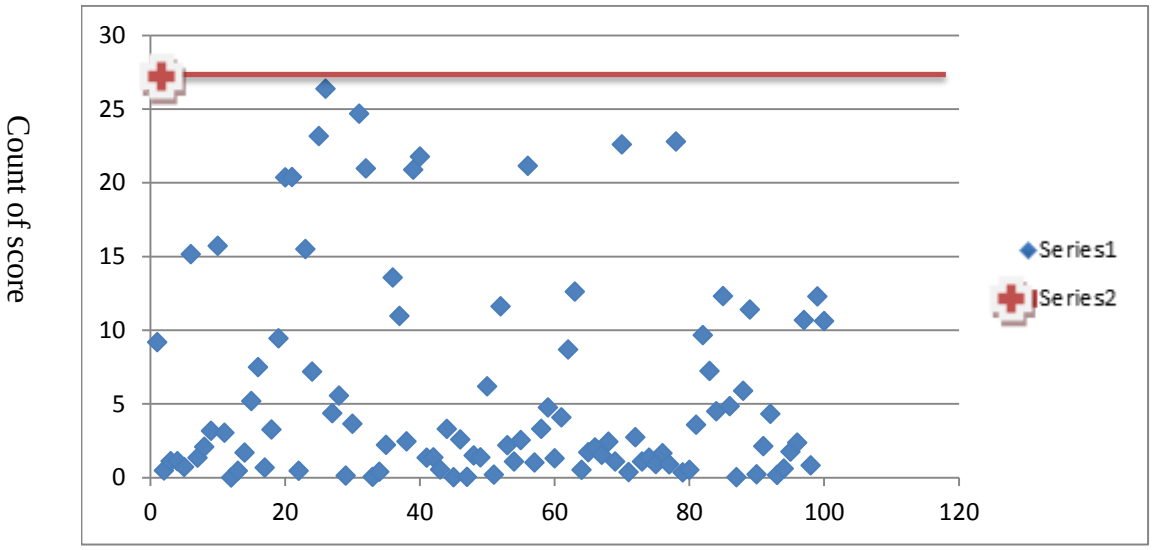
No. of scores

User 16



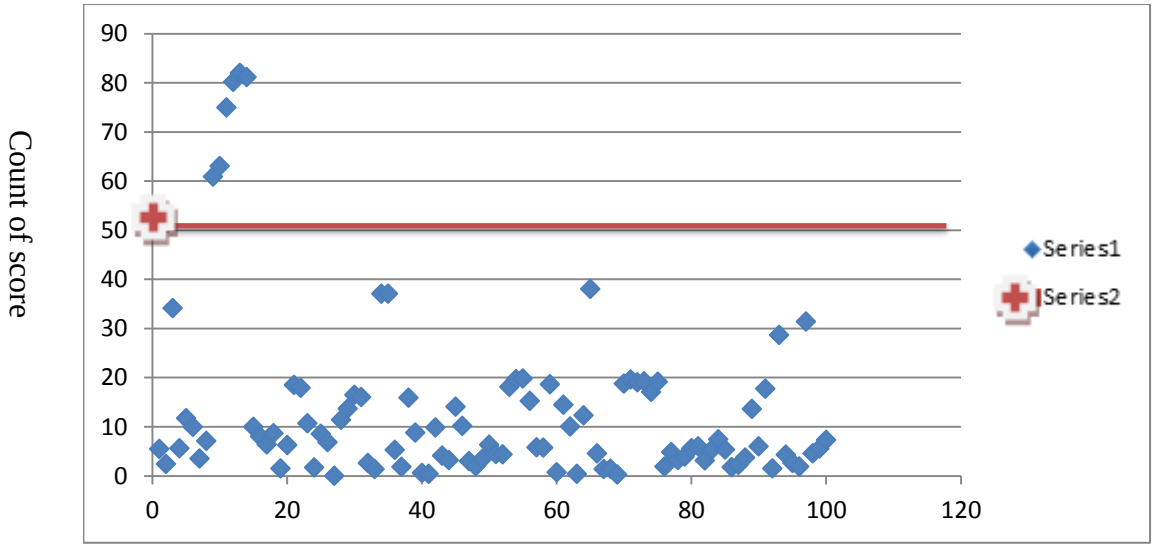
No. of scores

User 17



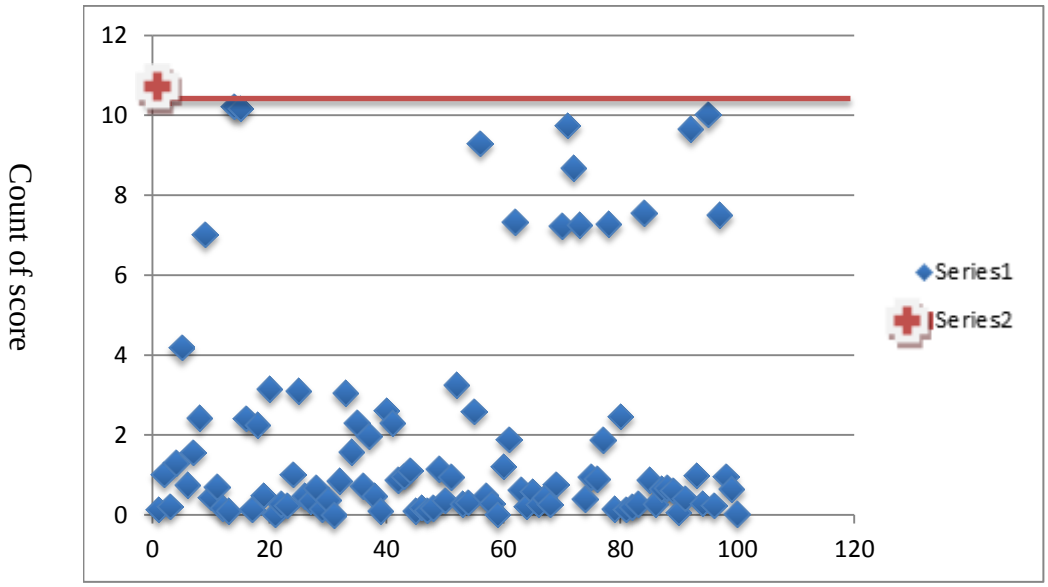
No. of scores

User 18



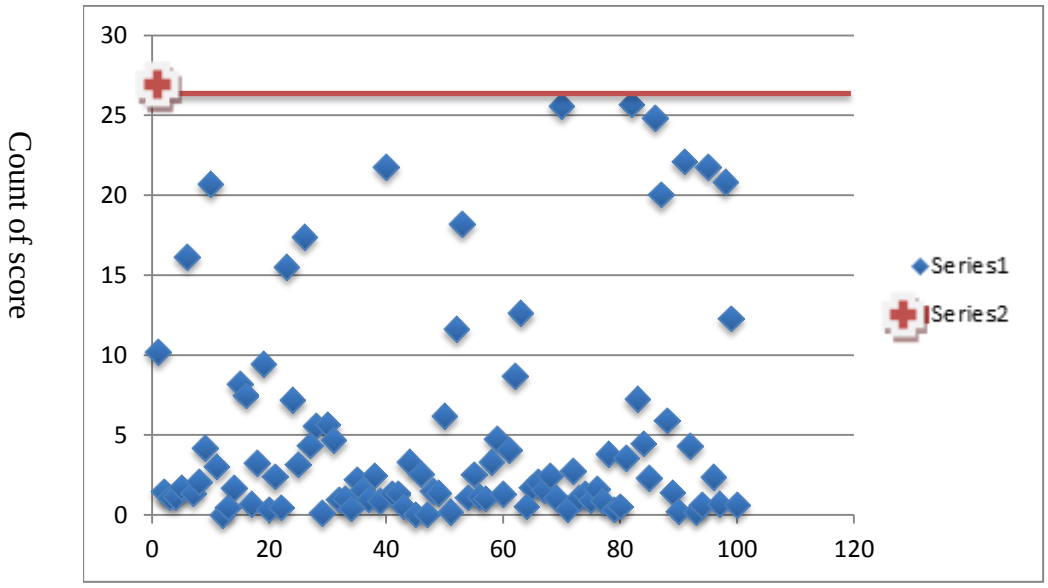
No. of scores

User 19



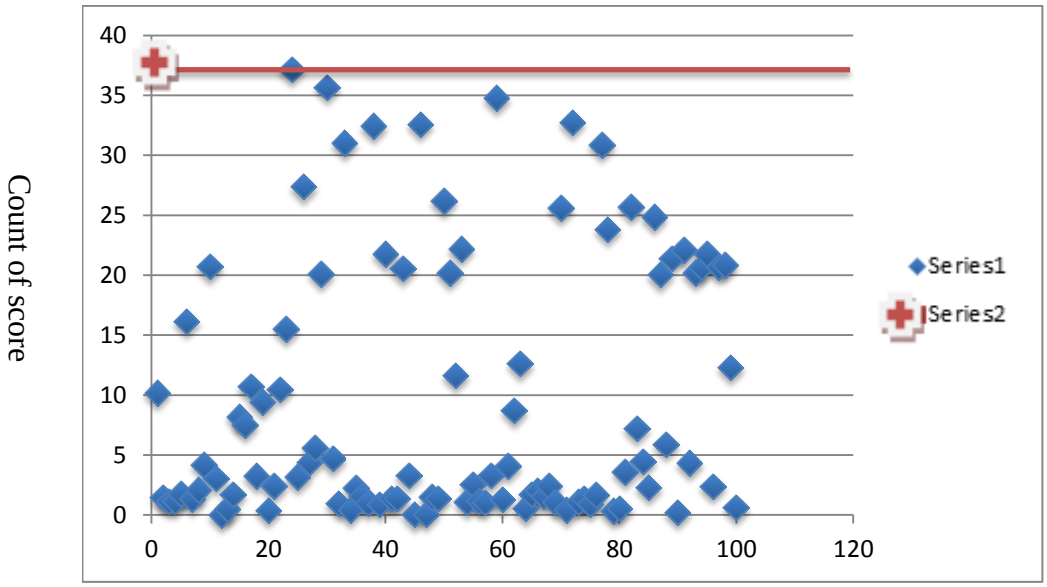
No. of scores

User 20



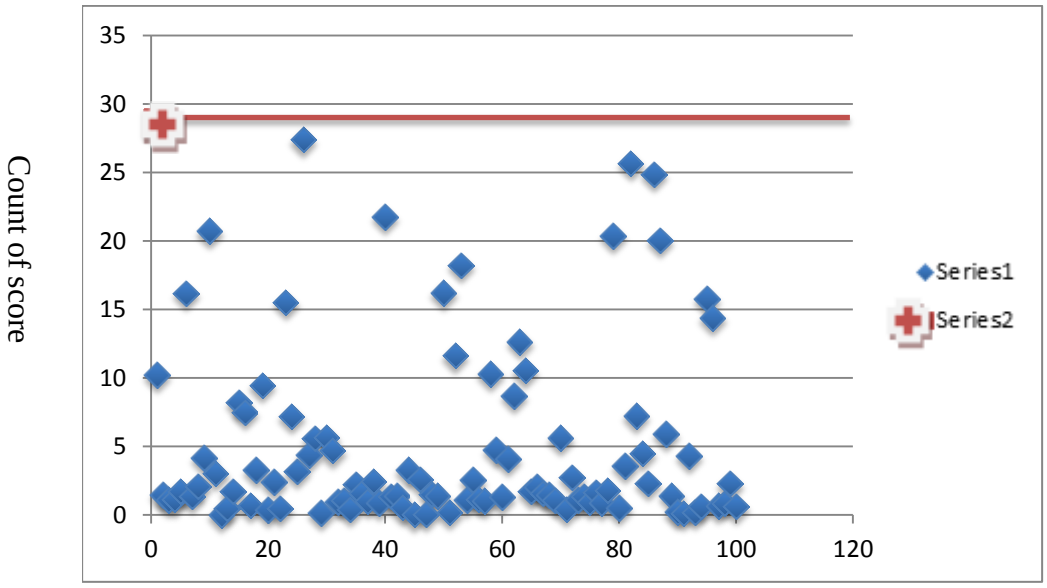
No. of scores

User 21



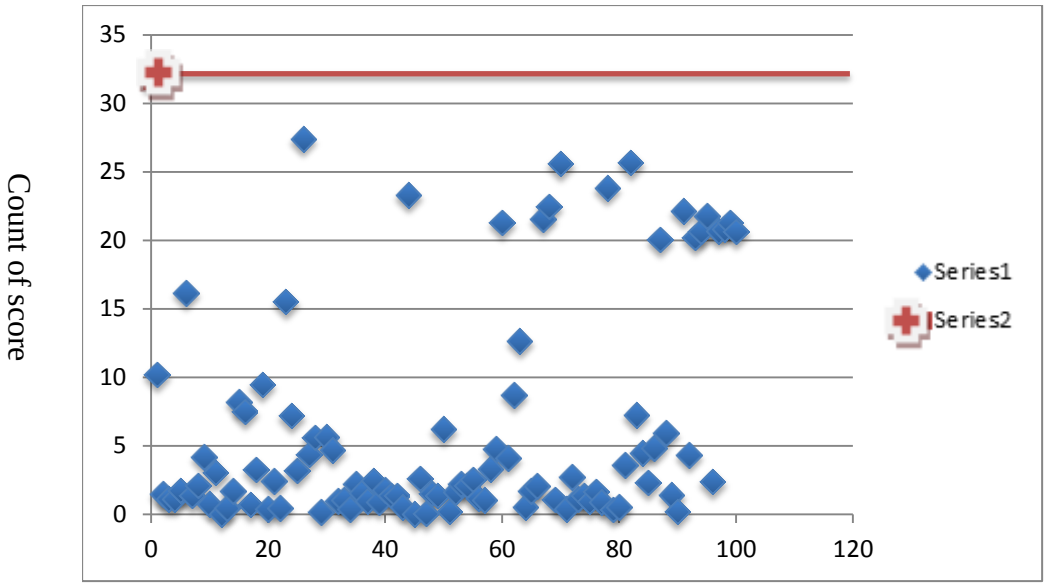
No. of scores

User 22



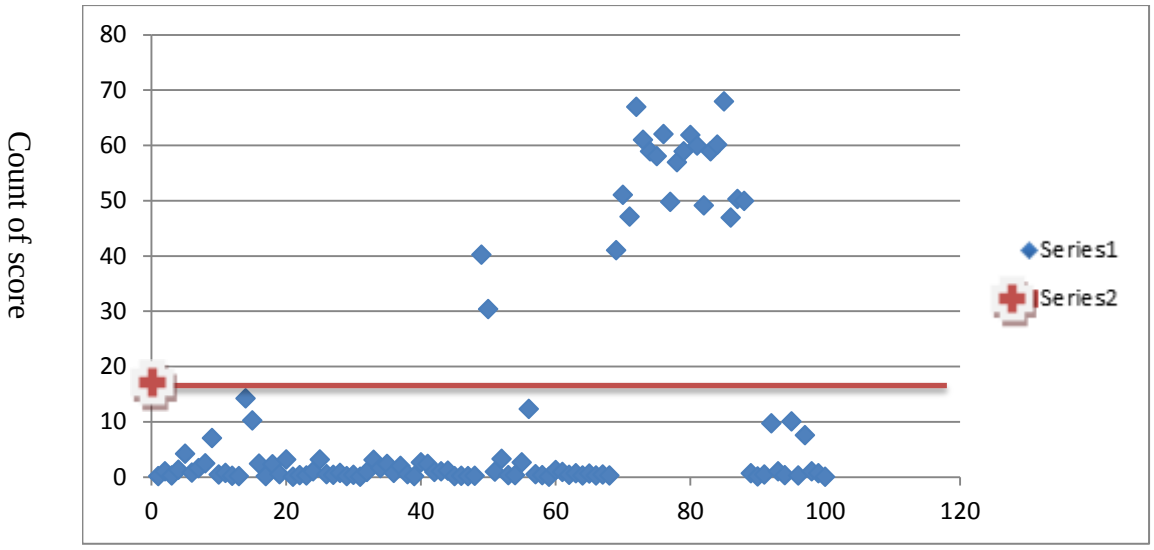
No. of scores

User 23



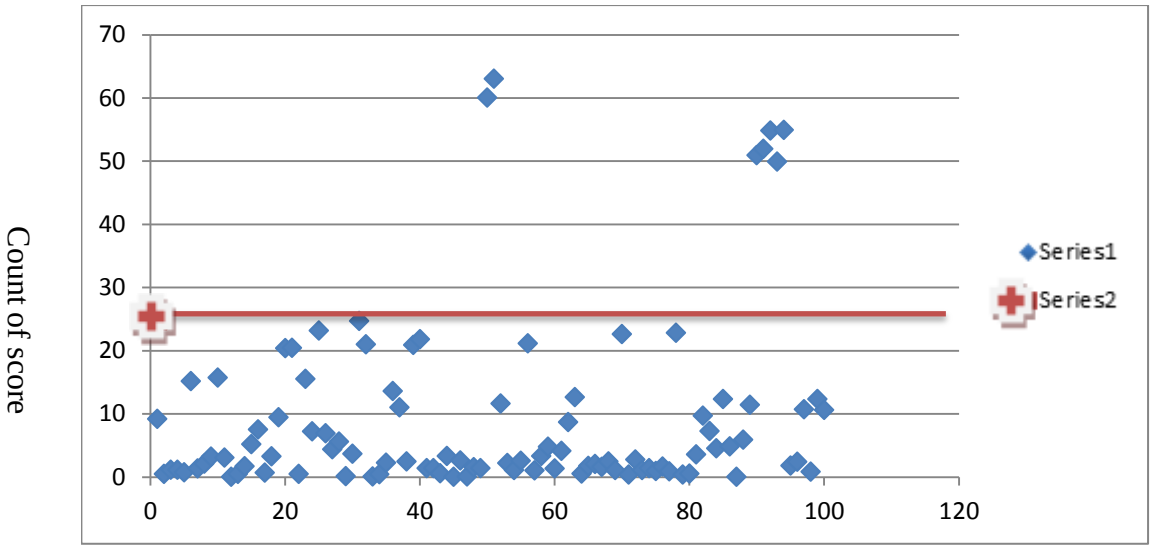
No. of scores

User 24



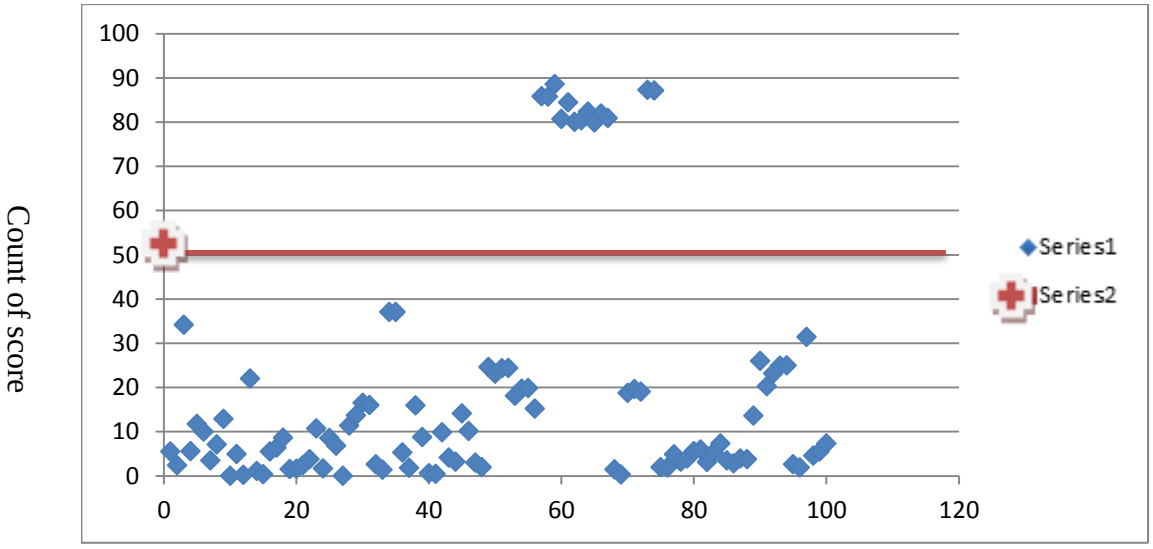
No. of scores

User 25



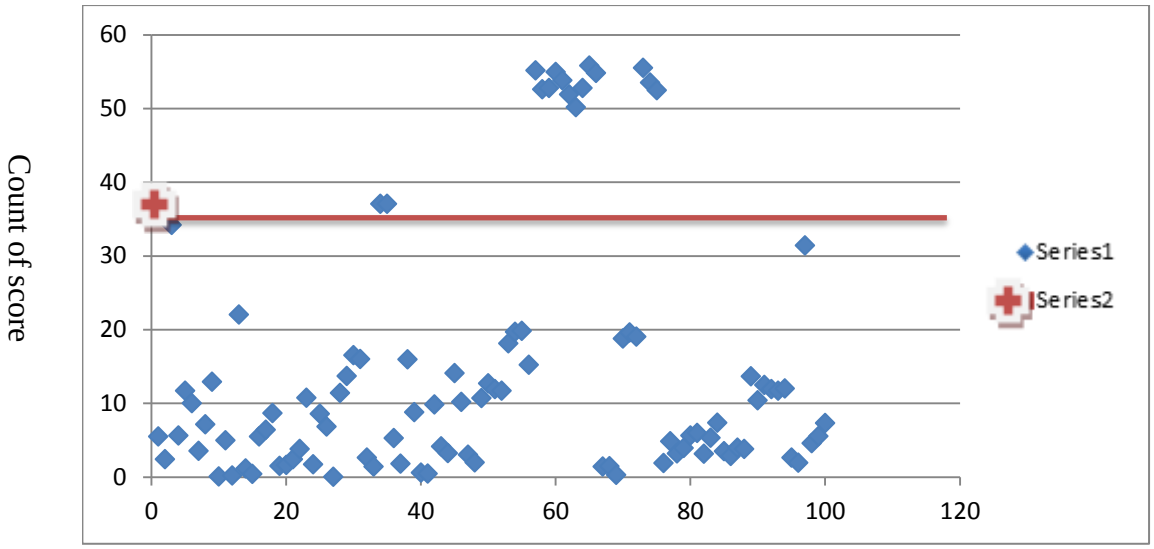
No. of scores

User 26



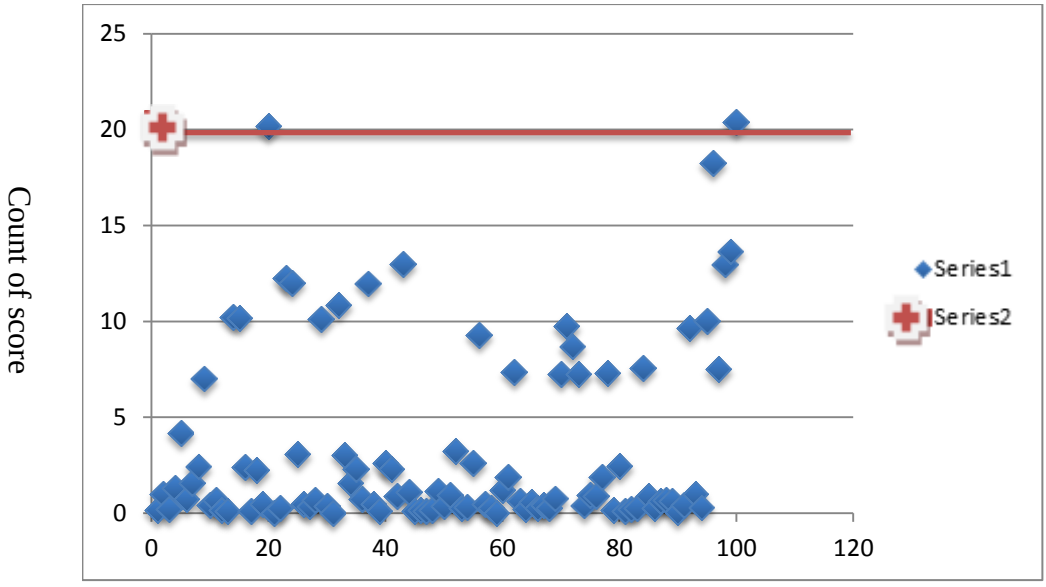
No. of scores

User 27



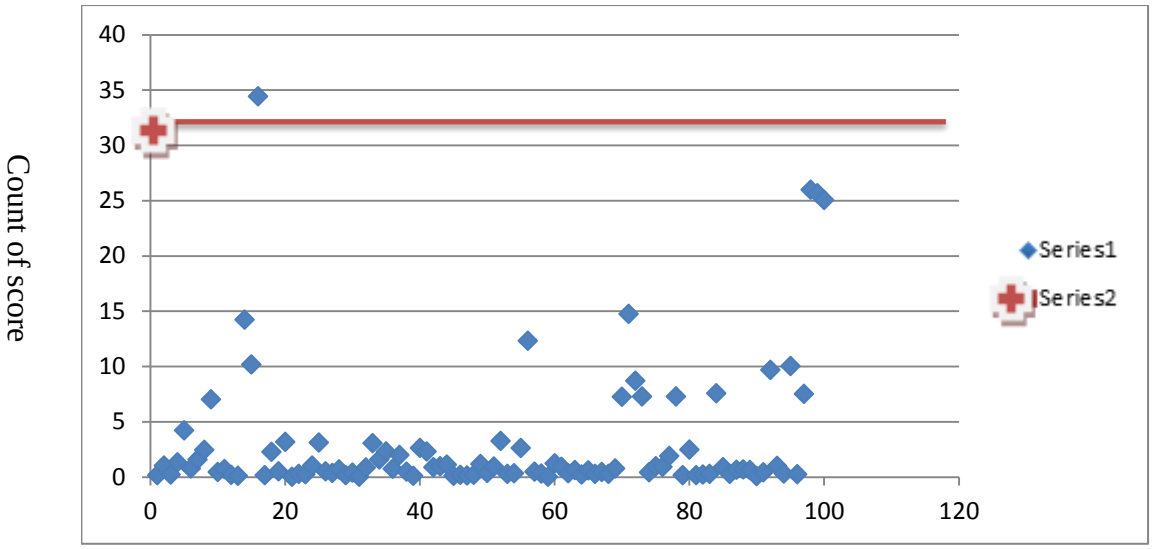
No. of scores

User 28



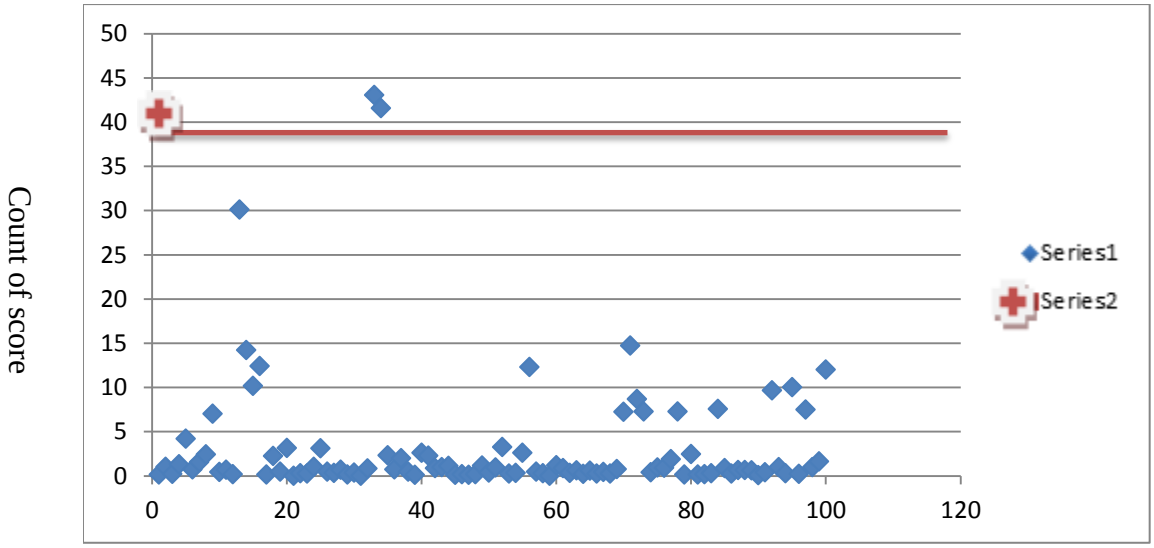
No. of scores

User 29



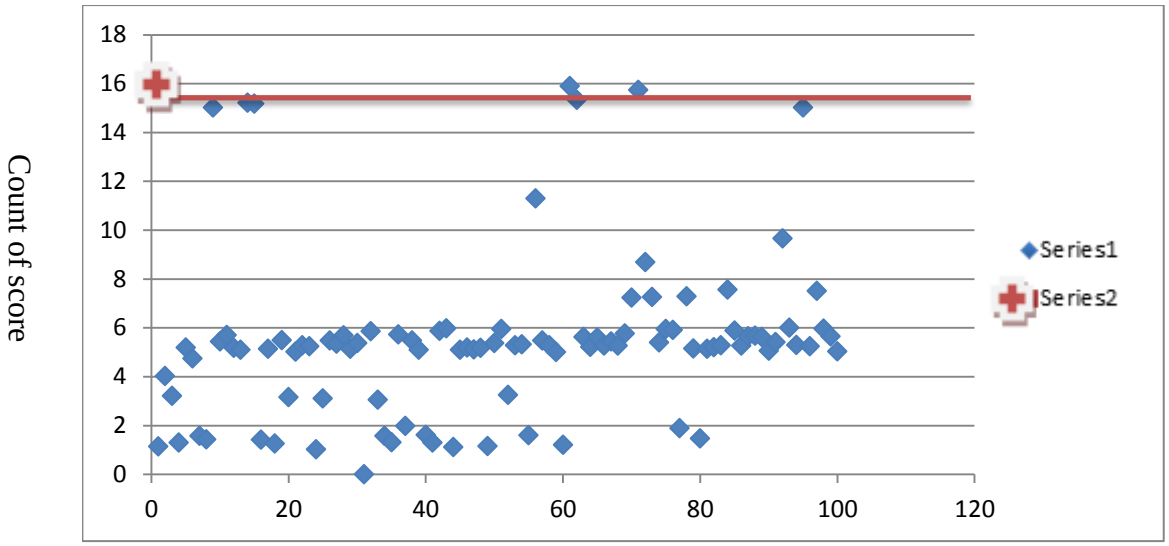
No. of scores

User 30

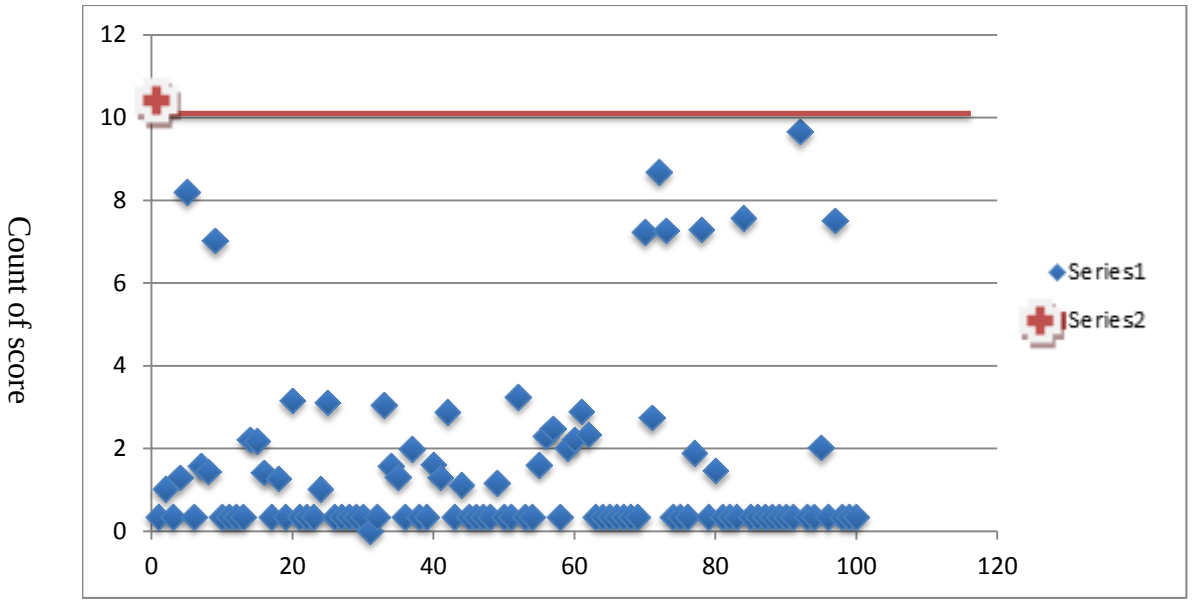


No. of scores

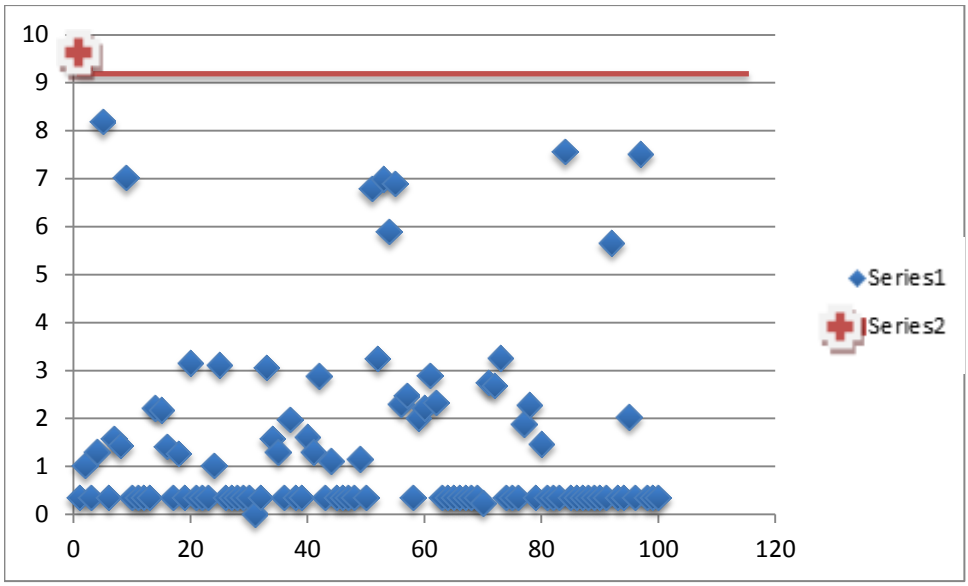
User 31



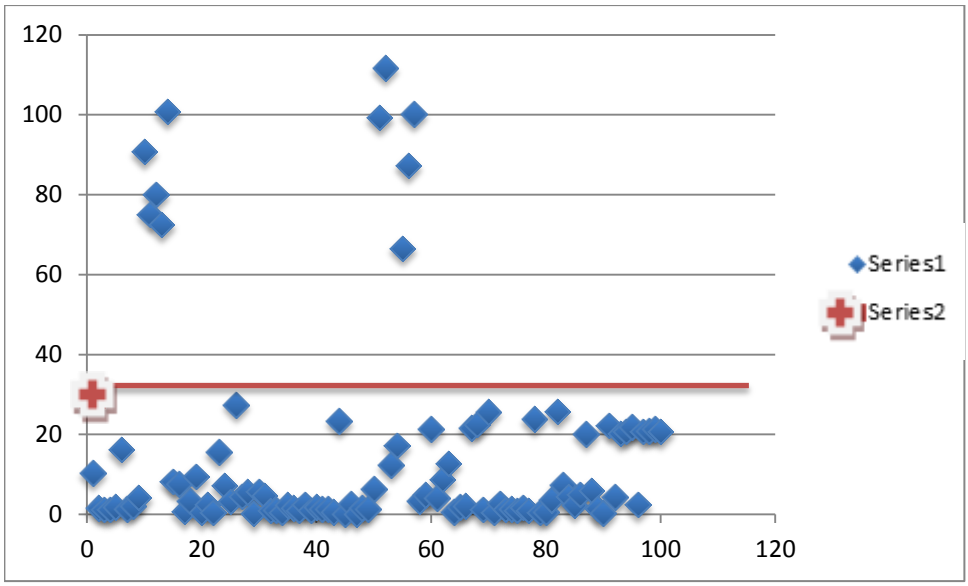
User 32



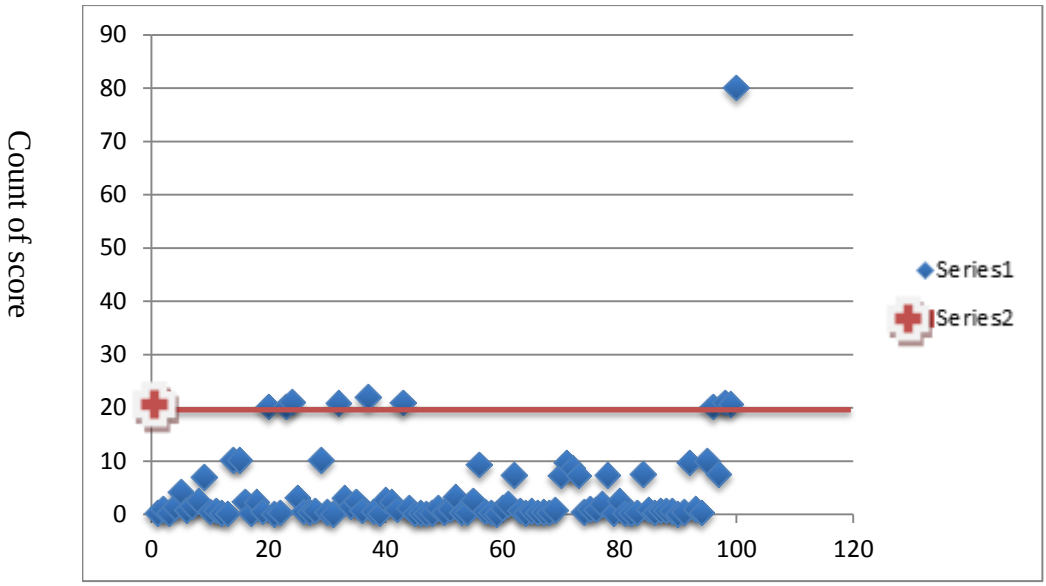
User 33



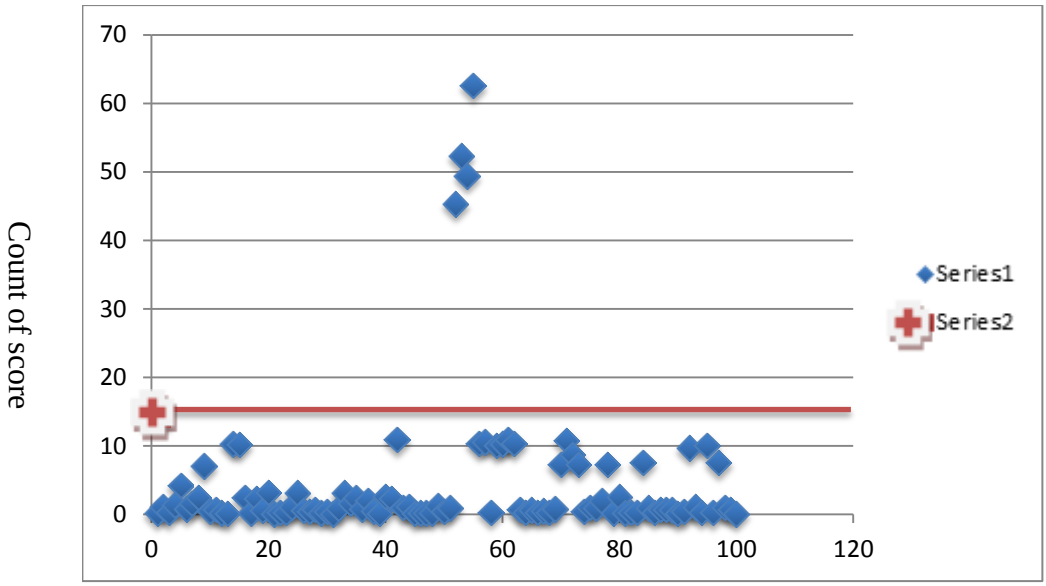
User 34



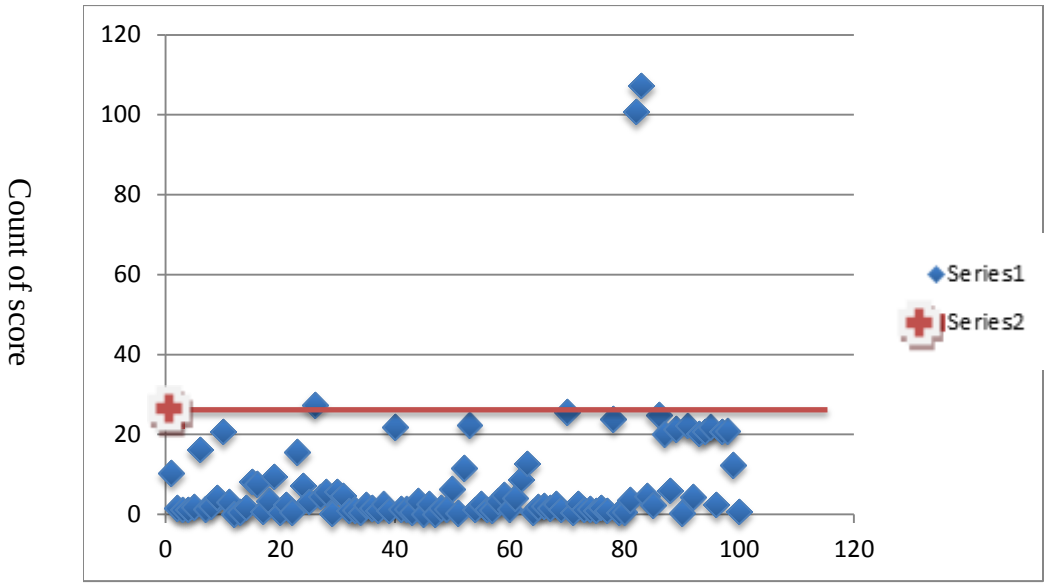
User 35



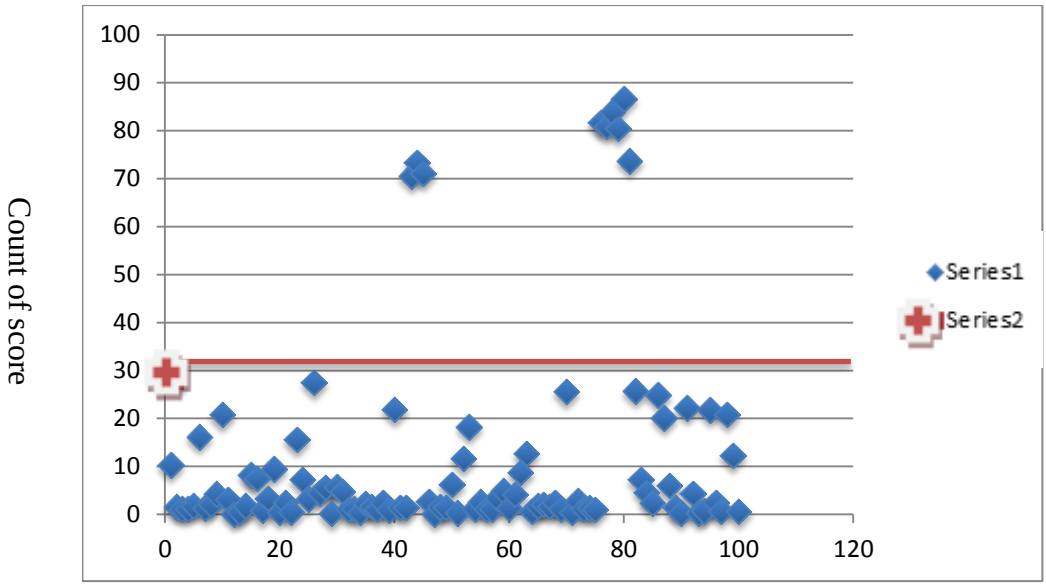
User 36



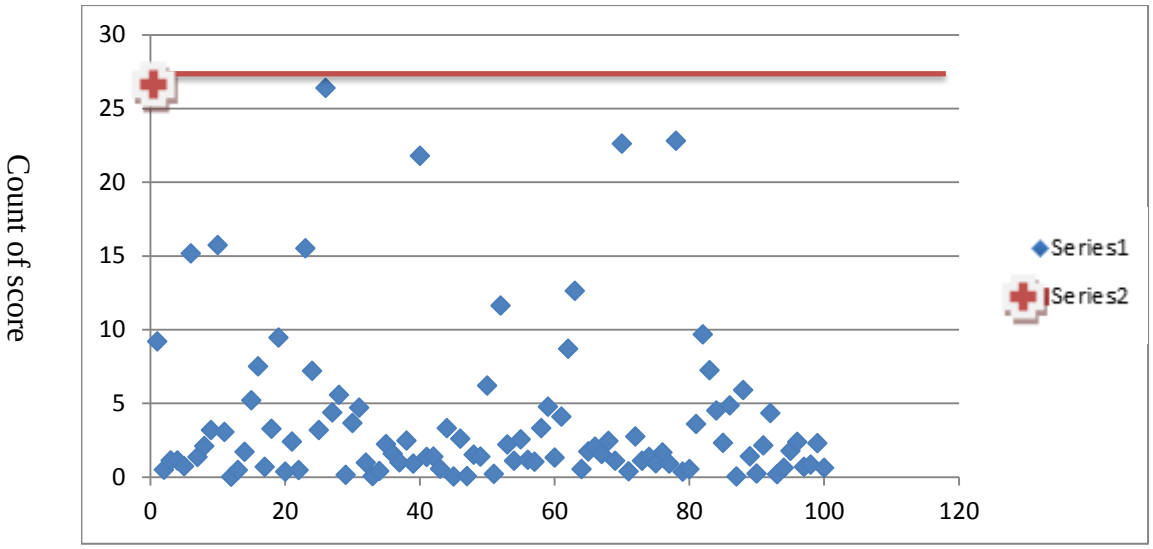
User 37



User 38

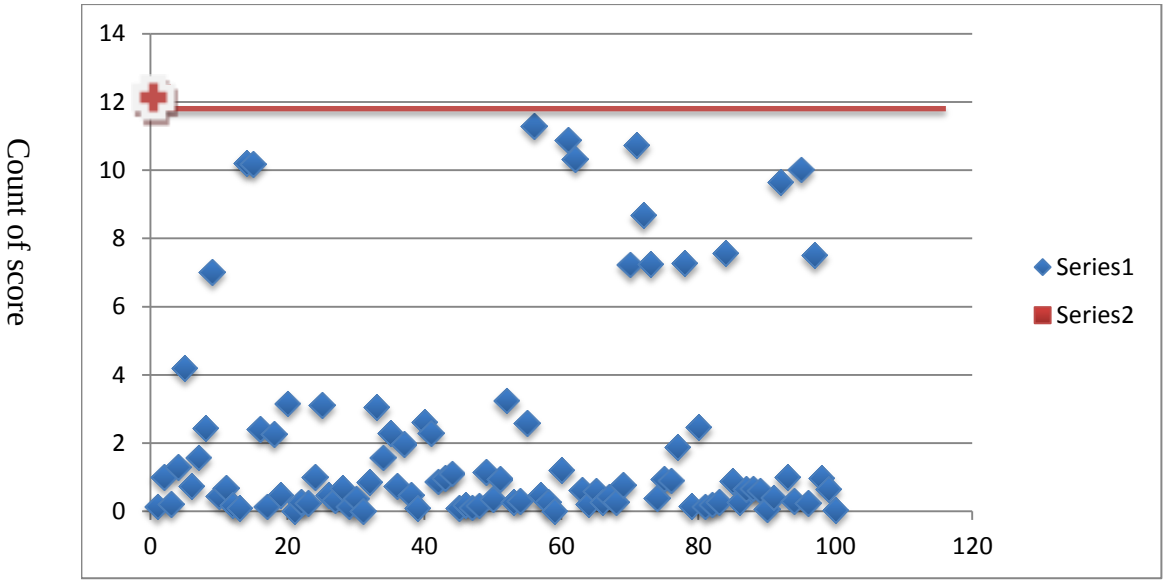


User 39



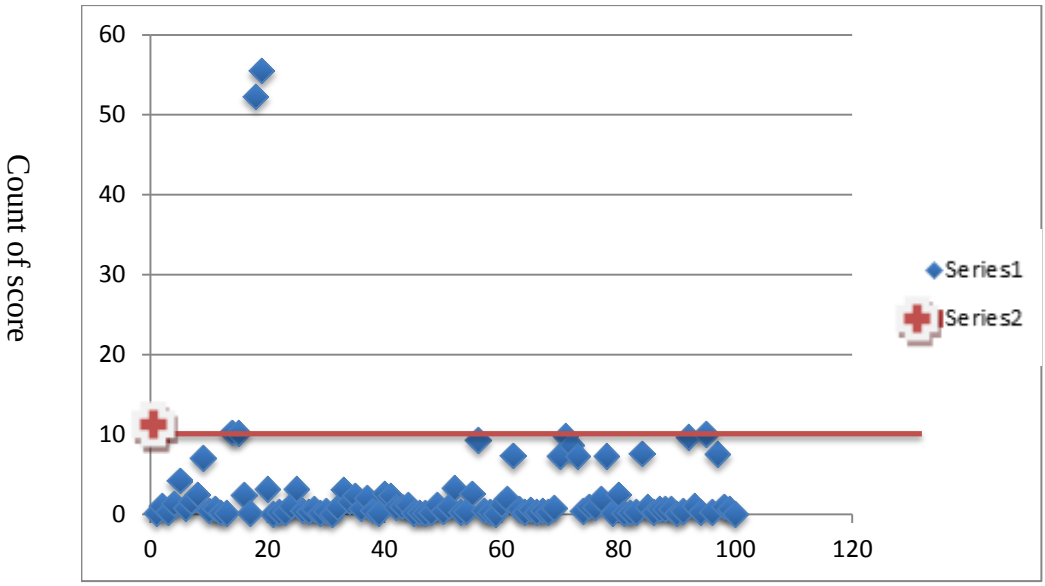
No. of scores

User 40

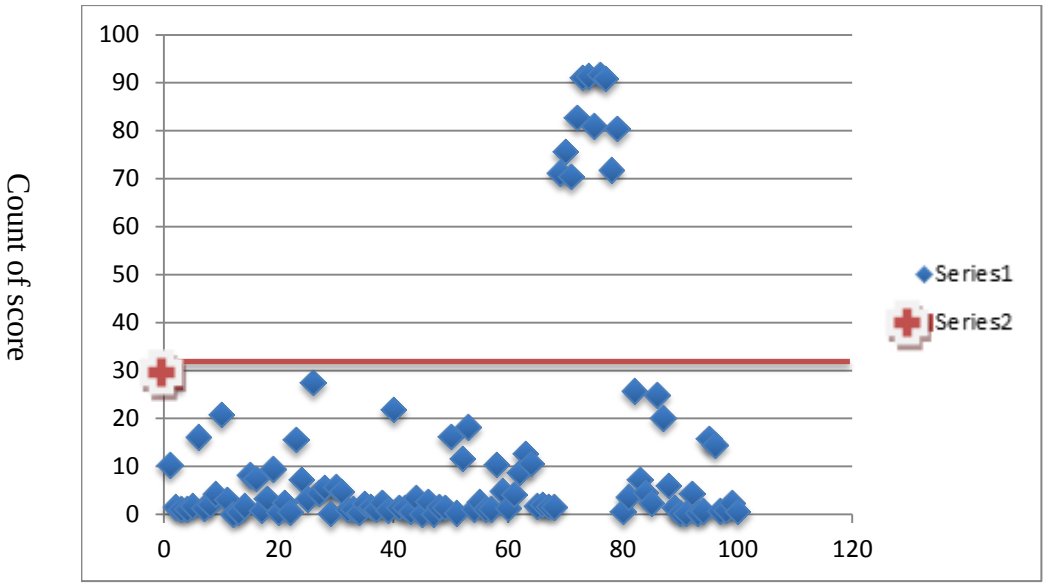


No. of scores

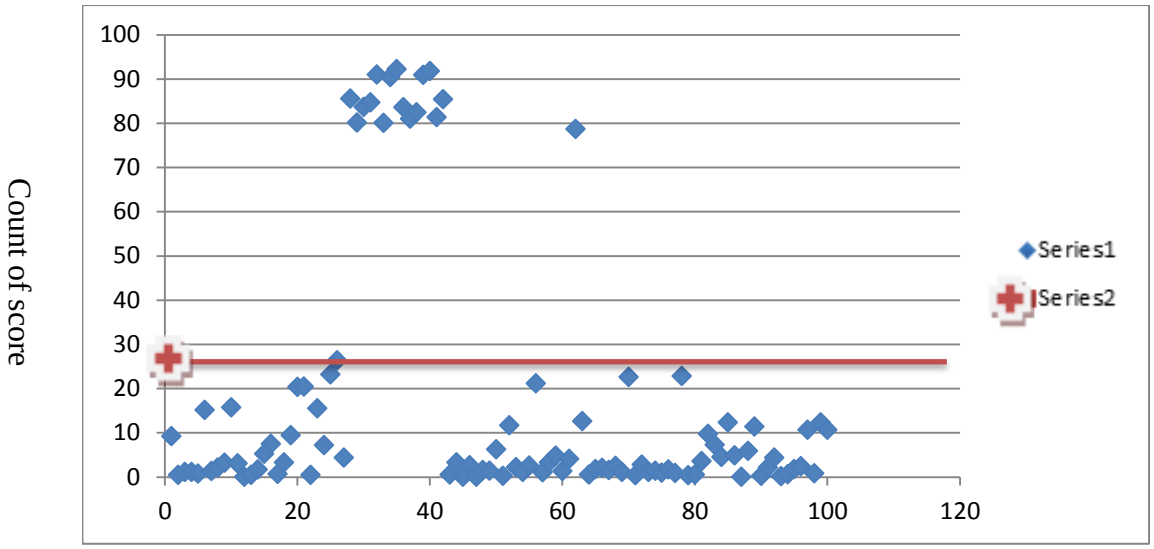
User 41



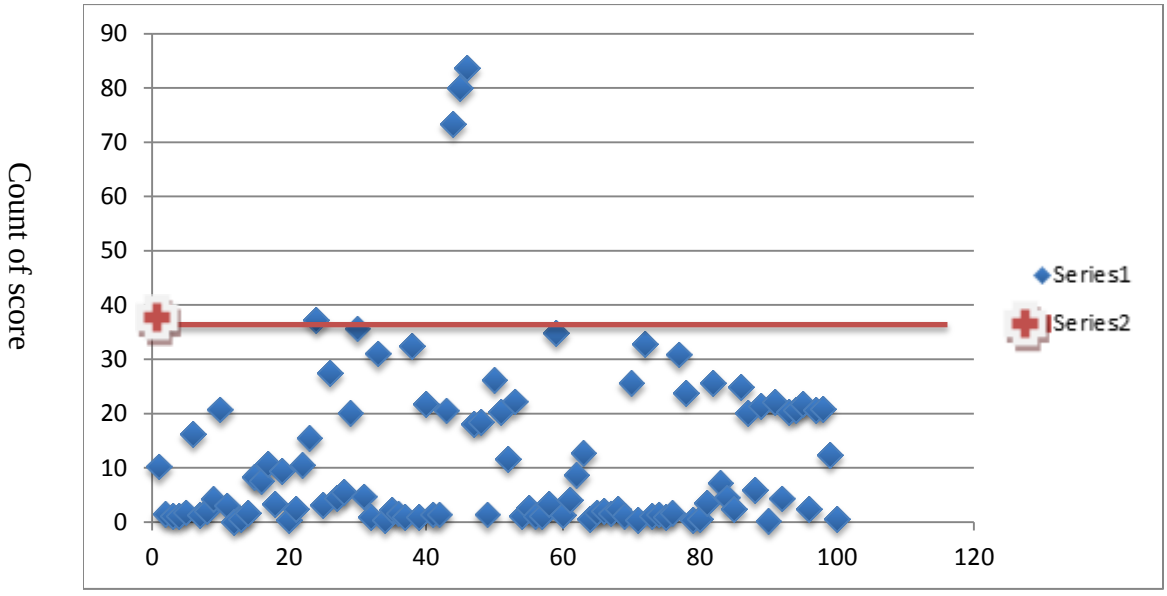
User 42



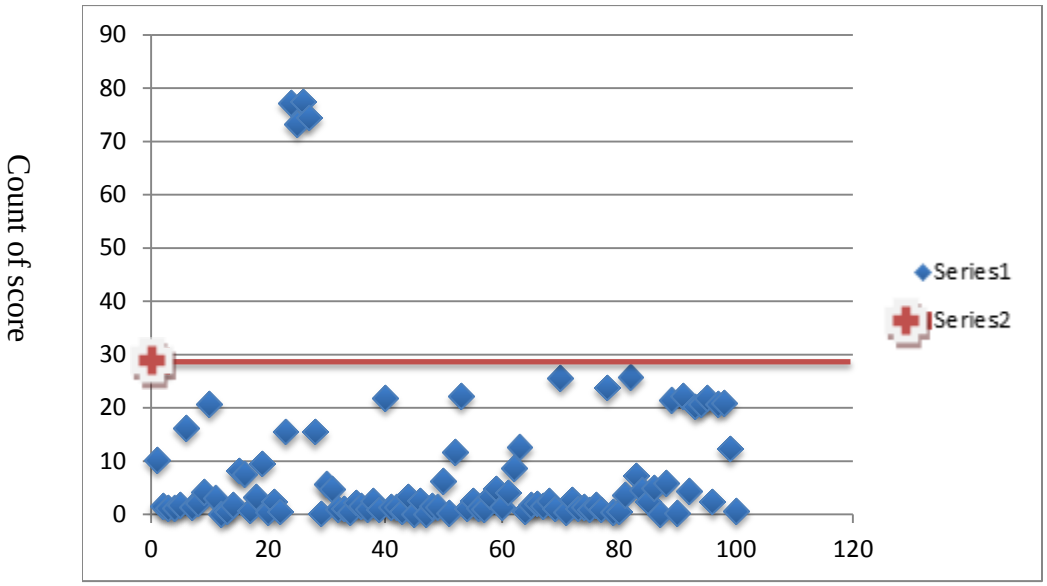
User 43



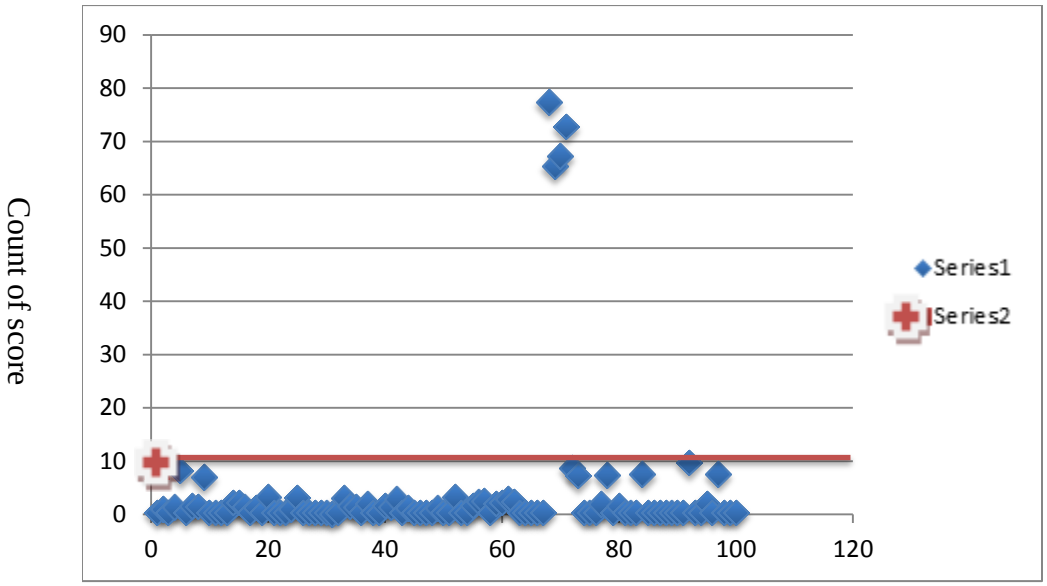
User 44



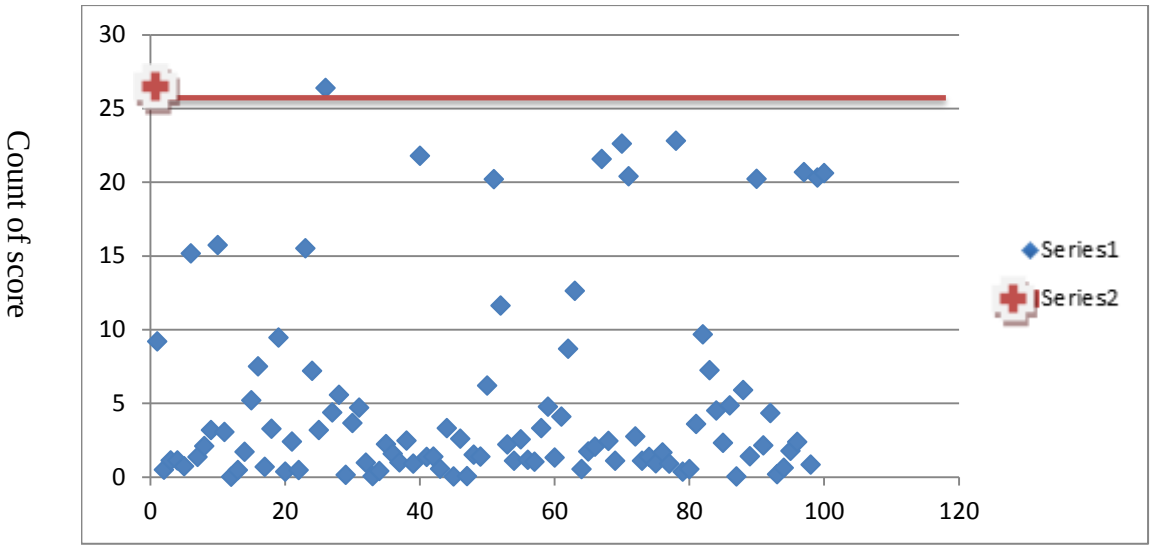
User 45



User 46

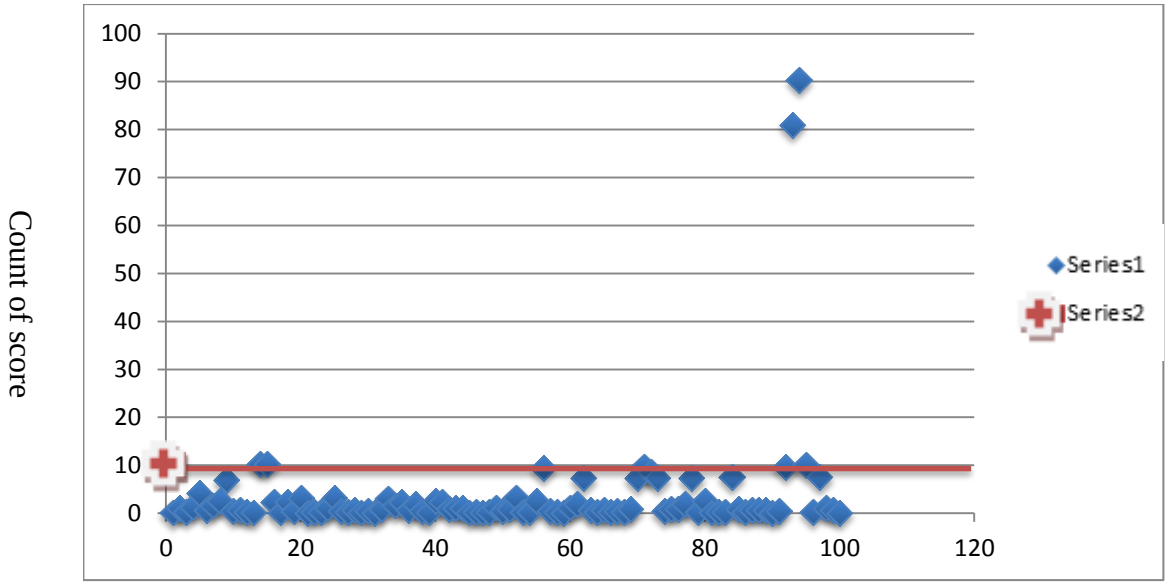


User 47



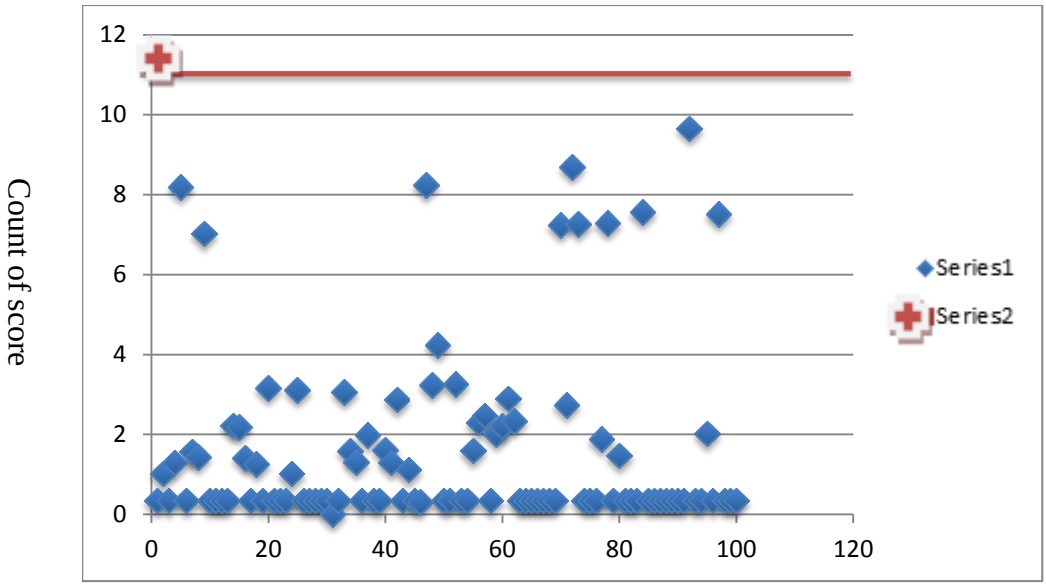
No. of scores

User 48



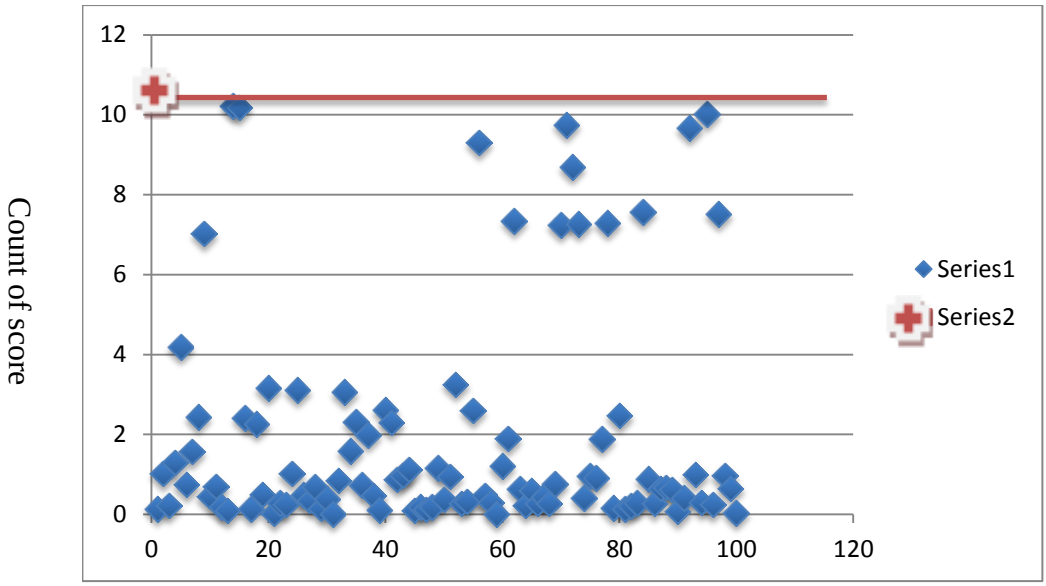
No. of scores

User 49



No. of scores

User 50



No. of scores