Continuous User Verification via Mouse Activities

Abstract—**Behavioral biometrics such as mouse dynamics are gaining attention these days to address the limitations of conventional verification systems. In this paper we present a novel method to continuously verify a user via their mouse activities. Our method, based on comparing against a simple statistical profile, was tested using 45 users and over 76,500 mouse activities. A total of 354,375 genuine and impostor verification attempts have been performed by deploying 175 different verifier setups. In our experiments, we achieved an impressive low Equal Error Rate (EER) of 6.72%. On average the EER was 13.42%. We opine that, our method can complement regular verification systems and can better serve for continuous verification purpose because of its simplicity.**

Index Terms—**Mouse Dynamics, Behavioral Biometrics, Continuous Verification.**

I. Introduction & Background

While relatively new, mouse dynamic user authentication does have existing research. The research already done, however, has limitations. The majority of the existing research was done with a relatively small population. Using a small population when conducting research can affect the results of that research. Larger sample sizes allow for more accurate results. Another limitation present in existing research is high false acceptance rates. When testing for authentication, the amount of false acceptances should be as close to 0% as possible. Having both higher false acceptance rates and false rejection rates leaves room for improvement on the research conducted. Research regarding mouse dynamic user authentication that has seen high success rates is when the authentication is used as a login measure, for example, using the mouse to draw a pattern to log into a machine instead of typing in a password or scanning a fingerprint. There are limitations to this research as well, the biggest being that drawing a pattern with the mouse can easily be reproduced. Also, this form of authentication is performed once and the user is granted access, it does not continuously verify that user.

One significant limitation in existing mouse dynamic user authentication research is the high false acceptance rates and false rejection rates. High FAR and FRR percentages mean that the system has low accuracy when successfully rejecting or accepting a user. We see this occurring with multiple accounts of previous research done in this area [1]–[3]. One of these research procedures only took into account mouse curves when authenticating a user [3]. This could be the reason they saw such high FAR and FRR rates, successful authentication is more difficult to achieve when only a few characteristics are used to build a user profile. In order to create a more practical and accurate system for mouse dynamic user authentication, FAR and FRR rates need to be significantly lower.

There has been some previous research done with very promising results. However, while this research saw false acceptance rates at only .43% and false rejection rates at 1.75%, the population size used was very low, they only used 11 subjects when conducting their research. While the results they achieved are more successful than the majority of other research done in this area, their low subject count raises questions about how reliable their system really is. When using a broader subject count, there is a possibility that these low FAR and FRR percentages would no longer be as low as originally recorded. It is hard to tell reliability when such a small population is being used. [7] One study that saw very low FAR and FRR rates at 0.55% and 3% respectively, only used 10 subjects to calculate these results. These subjects were also studied over a long period time, making the system much more accurate. This method would be difficult to implement on a larger group of people to get results that accurate [8]. Another instance where this research saw low FAR and FRR rates (just over 2%) also had a relatively low subject count. This research used a slightly larger number of subjects, 22, but even that number is still relatively low to see accurate results for large populations of people. [5]

Some promising research that has been done so far in this area used 37 subjects, saw relatively low FAR and FRR rates at slightly under 10%, and used a fast authentication time at 11.8 seconds [1]. Even this research could see improvement though. While the FAR and FRR rates are lower than most other research with a fast authentication time, they are still higher than what is desired. If lower FAR and FRR rates could be achieved at an even faster authentication time, practicality of the system would increase. Another instance of promising results found in this area of research saw an equal error rate of 1.3%. The researchers that conducted this research had an interesting point when discussing their results. They said that while their results were promising, they did not expect the results to stay as accurate if a larger sample size was used. They came to this conclusion based off of research done on keystroke dynamics, saying that larger sample sizes make chances of two users having similar characteristics increases significantly [10].

The most promising existing mouse dynamic authentication research achieved false acceptance rates at 0% and false rejection rates at 0.36%. The results achieved in this research make it the first conducted on mouse dynamics as a continuous verification system to reach the European standard for access control, which is a FAR rate less than 0.001% and a FRR rate less than 1% [6].

One method of mouse authentication that has seen very promising and positive results is using the mouse as a login procedure. One research procedure had users use the mouse to login as if unlocking a combination lock. They saw successful results with low false rejection and false acceptance rates [4]. Another research procedure had users draw a complex figure with the mouse to login. They saw a 93% successful verification rate with their research [9]. While the results of mouse authentication through the use of a mouse to draw a pattern to login into a system are promising, it is not as practical as a continuous verification system would be. Drawing a pattern with the mouse is more easily replicated than a users specific and unique everyday normal use of the mouse.

Our research on mouse dynamic user authentication addresses the limitations talked about. We have used a larger population base to take our samples from to increase our accuracy. With our research, we have taken a larger variety of mouse characteristics into consideration when measuring the uniqueness of a users mouse characteristics. Using a larger amount of mouse characteristics can build a more complete and specific profile for each user. Contribution of this work follows:

- We propose a simple method of user verification using mouse dynamics. Unlike existing methods which rely on sophisticated calculations, our method uses basic statistical information. A light weight system such as ours, is more suitable for continuous verification in real time.
- We performed 354,375 rigorous tests under 175 experimental setups. Our dataset containing mouse activities collected from 45 users, was large enough for a sustainable test results. Our system, showed impressive results with EER as low as 6.72%.

We organized the paper as follows. In Section II we discuss data collection. In Section III we present the feature extraction method. In Section IV we present our method of continuous verification. In Section V we discuss the experimental setup. In Section VI we present our results and analysis. We conclude in Section VII.

II. Data Collection

We wanted our data set to include movements that were natural to best simulate day to day actions a user would perform on the computer. This is to have a more relevant set of data that our authentication method would see in a more practical environment. To do this, we asked a class of students to record their mouse movements. The students were instructed to perform any activity on the computer they normally would for an hour period. In order to log the mouse activity of users we created a simple mouse logger. The logger was a small piece of software that ran on the windows operating system. The logger, when run on a machine, logged the position of the cursor using the windows method GetCursorPos(), the state of the left and right mouse buttons (up or down), and a timestamp every 15.6ms. 15.6ms being the average default timer resolution that windows supports. The resulting log was then placed into a text file. We then screened the samples and removed samples with very few (0 to 10) movements and samples that weren't an hour long. Post-screening we had 45 valid user samples. Around 76,500 mouse activities have been processed for our experiments. Our samples consisted of university students collected from late November 2014 to early December 2014. Table I shows some statistics of collected samples.

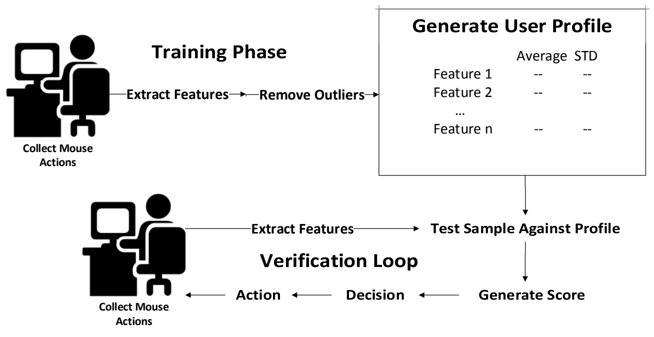


Fig. 1: An overview of continuous verification with mouse movements. In the training phase a user creates a profile based on their mouse movements (The average and standard deviation of each mouse feature). The profile is used to represent a user in the verification phase where features from live mouse movements can be verified against the profile in order to determine if the live mouse movements came from the user who created the profile. If the verifier determines that the live mouse movements are illegitimate an action can be taken. (e.g. logging the user out, creating a log of the incident, etc.)

Activity	Maximum	Minimum	Average
Left Clicks	716	37	547.84
Right Clicks	690	0	122.8
Double Clicks	1282	1	181.64

TABLE I: Overview of the samples collected

III. Feature Extraction

In order to classify a test sample we needed some way to compare test samples and training samples. To do this we used features that are found in our definition of a mouse movement. The features, defined in Table II and visualized in Figure 2, were found to be unique among users. In total we had 6 features and 3 sub-features. The double click length feature is interesting in that it contains 3 sub features allowing each sample to have a unique double click feature even if the entire length of the double clicks happen to be the same among users.

We defined a mouse movement as a period of cursor movement followed by a click. This isn't to say that a click without any prior movement is not considered a mouse movement. It would be a mouse movement with zero speed, acceleration, and jerk. If the cursor sustains no action (movement or clicks) for a period of 500ms or longer we classify that period and any preceding cursor movement as a pause. This allows for periods where a user is not concentrating on moving the mouse with intent while not having it effect periods where a user is moving the mouse with intent. We do not use pauses for classification in our method. If a mouse movement ends in a pause instead of a click we define that as a partial mouse movement. We don not consider these partial movements in our classifier because their occurences in the samples are too sparse. While they may hold unique identifying information about a user its more difficult to tell if the partial mouse movement came from purposeful movement (i.e. shaking the mouse to wake up the screen) or accidental movement (i.e. knocking into the mouse with your elbow.)

IV. Methodology

A. Outlier Filtering

For our setup we first went through each of the user samples and extracted each of the features, as defined above, from each mouse movement. To remove rare cases of mouse movements where the user provides a mouse movement far from the normal, we implemented an outlier removal scheme. We define a mouse feature to be an outlier if it is more eccentric than 66% of the population of the same mouse features for a single user. For example, if we are looking to find if a particular mouse feature M is an outlier:

Algorithm 1 Determines if a single feature F is an outlier from the rest of the population of its own kind. $featurePopulation$ is an array of same-typed features from a testing sample that F is also a member of.

```
1: procedure IsAnOutlier($F$)
2:     count ← 0
3:     for each $feature$ in $featurePopulation$ do
4:         if $|F - feature| > R$ then
5:             count ← count + 1
6:     if $count \div featurePopulation_{size} \geq .33$ then
7:         return True
8:     else
9:         return False
```

We defined the variable R, for each mouse feature, a number large enough to include most of the population while small enough to remove the noise from the data. We found optimal values for our R values after extensive experimentation we will describe later on.

We created a simple algorithm for determining if a particular record of a mouse feature is an outlier outlined in Algorithm 1. For a given set of mouse features from a training sample (the featurePopulation), we compared each feature in the featurePopulation against every other feature of the same type, i.e. a Single Left Click Length feature is only compared to every other Single Left Click Length feature. Then a simple tally is kept of how many of the features in the featurePopulation have a distance greater than R as shown on line 4 & 5 of Algorithm 1. On line 6 we simple restate that if the if the tally is greater than 33% of the featurePopulation we claim the feature F is an outlier and discard it from the training set.

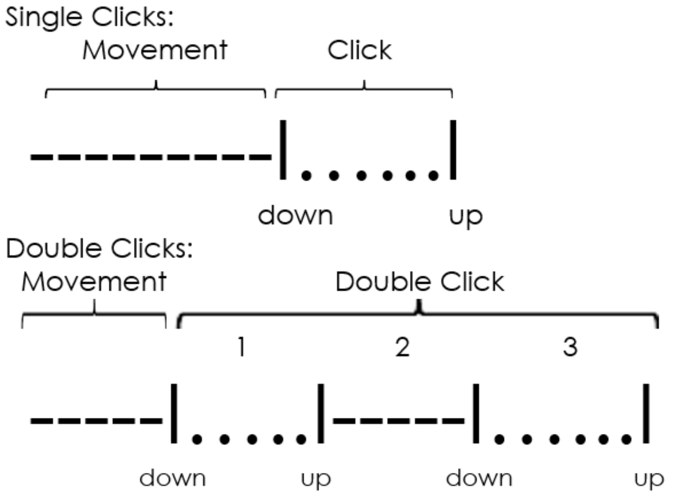


Fig. 2: A visual representation of the click features. Dashes representing no button activity, dots representing periods where the mouse button is down, and vertical bars representing the first record that the mouse button has changed state.

B. Verification

To authenticate a user based on their mouse movements we analyze six mouse features over the course of a sample. Our method is a simplistic user-authentication model. We create a profile based on a user's past behavior then test further samples against that profile to determine if the sample we test with is close enough to the previous profile to be considered the same user. We then split the edited list of features into a testing set and a training set. For each file we create user profiles based on half of the user's sample (about 50 mouse movements on average for our samples). The profiles being simply the averages and standard deviations of each of the mouse features. The mouse movements from the sample that we did not use for the profile go into the user's testing set. Using a simple classification scoring method we can see how well a user profile and a testing set of data match up. For our entire set of data we took every user profile and tested each against every testing set. Our custom testing method generated a score for each test between 0 and 1 based on a simple scoring formula $score = \frac{NumberOfAcceptedFeatures}{TotalNumberOfFeatures}$. 1 being a perfect match and 0 being a perfect mismatch. It should be noted we accept a particular feature if $Average_{Training} - \text{M} * STD_{Training} \leq featureValue \leq Average_{Training} + \text{M} * STD_{Training}$. We tested by comparing each of the mouse movements in a testing set against the users profile. If the features in the

Feature	**Description**
Single Left Click Length	The length of a click from the first record the mouse button is found to be down to the first record the mouse button is found to be up.
Single Right Click Length	
Double Click Length	Split into 4 sub-features: Total Length and intervals 1,2, and 3. Total length: The time between the first record the mouse is found to be down and the first record the mouse is found to be up after the second click. Interval 1: The first single left click length in a double click. Interval 2: The time between the first record of the left mouse button being found up after the first click to the first record of the left mouse button being found down for the second left click. Interval 3: The second single left click length in a double click. Note: A visualization of the intervals in Fig. 2
Speed	Distance the cursor travelled divided by the total time of the movement up until the click.
Acceleration	Change in Speed over time
Jerk	Change in Acceleration over time

TABLE II: Descriptions of the features used

mouse movement are within M standard deviations of the corresponding means in the user profile then the test gets a point. The total amount of points accrued is divided by the total amount of mouse movements in the file to generate the score.

V. Experimental Setup

We had a few parameters where it wasnt immediately apparent how to set. For the R variables we needed values that can discriminate mouse feature values that are too eccentric while keeping as much unique data as possible. To solve this we ran a script that ran the tests and found the EER over an array of values for each parameter to determine the optimal values where the EER is at its lowest. We found that an M value of 1.9, a Single Click R of 15, and a Double Click R of 175 to produce the lowest EER for our set of data.

Parameter	**Values Tested**
Single Click R Value	15 20 25 30 25
Double Click R Value	175 180 185 190 195
M Value	0.7 1 1.3 1.6 1.9 2.2 2.3

TABLE III: The values used for our parameters for testing. By varying the parameter value we experimented with 175 (5 Single Click R x 5 Double Click R x 7 M Values) different experimental set-ups. Note that we do not use any R parameter for the other features since they show less variance. In total, we experimented with 354,357 verification attempts (45 users x 45 users x 175 experimental set-ups).

VI. Experimental Results & Analysis

To calculate the results of our experiment we took each users profile and tested that against every other set of testing data. Therefore, for experimental setup, we generated 2025 (45 $\times$ 45) verification scores. We then calculated the False Reject Rate (FRR) and Impostor Pass Rate (IPR) values for varying threshold values from 0 to 1 with a stepsize of 0.01. We plotted 1000 FRR and IPR values as shown in each subfigure in Figure 3. For space constraint, we show FRRs and IPRs for four experimental setup out of 175 setups. In each subfigure, the EER is maked as the crossover point between FRR and IPR curves. In total, we generated 175 EERs from all setups. In Table IV, 35 EER values generated by varying double click R and M values when single click value set to 20ms, are listed. In Figure 4, EER values are shown for four other single click R values. From the figures, our observations are as follows:

- The Lowest EER achieved is 6.72% (see Figure 4(c)) when M= 2.2, single click R=20ms and double click R=185ms. Average EER is 13.19% and standard deviation is 5.77%.
- Among the parameters, M is found to have greater influence on accuray (see the curves in subfigure in Figure 4). This urges to explore setting up the M value for each user seperately rather than generic one.
- Single click R value also show some level of influence on accuray. EERs in Figure 4 (b, c) are lower than EERs in Figure 4 (a, d). It is worth mentioning that, number of single clicks are found to be far more than other activities.

		R **values**				
		175	**180**	**185**	**190**	**195**
M **values**	**.7**	14.86%	10.53%	9.33%	9.33%	8.99%
	1	30.02%	24.23%	24.23%	27.37%	24.00%
	1.3	14.46%	13.90%	10.40%	13.56%	13.33%
	1.6	11.35%	10.40%	10.40%	10.59%	11.81%
	1.9	13.35%	12.00%	12.00%	11.81%	11.36%
	2.2	13.36%	6.77%	6.70%	6.72%	6.88%
	2.5	7.71%	8.00%	8.00%	7.89%	7.50%

TABLE IV: The EER rates found across multiple parameter values. Each column is a separate double click R value while each row is a separate M value. The entire table is tested against one single click R value set to 20ms.

VII. Conclusion

Our comprehensive experiment which comprises X thousands verification attempts carried over 45 users, shows the effectiveness of our method. On average, the EER we found was 13.42%. The lowest EER was found to be 6.72%. In future, we want to test our method on more user samples. To improve accuracy even further, we want to set the parameter values (R and M) for each users separately instead of adopting generic values. We believe that, a simple system with impressive accuracy such as ours, can easily be considered for continuously verify system users in real time.

Acknowledgment

We acknowledge the acknowledged acknowledgees.

References

[1] Chao Shen, Zhongmin Cai, Xiaohong Guan, Youtian Du and R. Maxion, 'User Authentication Through Mouse Dynamics', IEEE Trans.Inform.Forensic Secur., vol. 8, no. 1, pp. 16-30, 2013.

[2] Z. Jorgensen and T. Yu, 'On mouse dynamics as a behavioral biometric for authentication', Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security - ASIACCS '11, pp. 476-482, 2011.

[3] D. Schulz, 'Mouse Curve Biometrics', 2006 Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference, 2006.

[4] H. Jahankhani, D. Palmer-Brown and K. Revett, 'A Survey of User Authentication Based on Mouse Dynamics', Global E-Security. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008, pp. 210-219.

[5] A. Ahmed and I. Traore, 'A New Biometric Technology Based on Mouse Dynamics', IEEE Trans. Dependable and Secure Comput., vol. 4, no. 3, pp. 165-179, 2007.

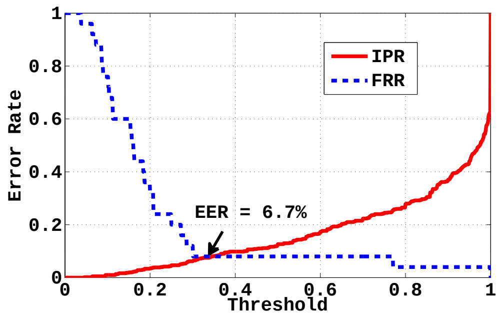
[6] Y. Nakkabi, I. Traore and A. Ahmed, 'Improving Mouse Dynamics Biometric Performance Using Variance Reduction via Extractors With Separate Features', IEEE Trans. Syst., Man, Cybern. A, vol. 40, no. 6, pp. 1345-1353, 2010.

[7] M. Pusara and C. Brodley, 'User re-authentication via mouse movements', Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security - VizSEC/DMSEC '04, 2004.

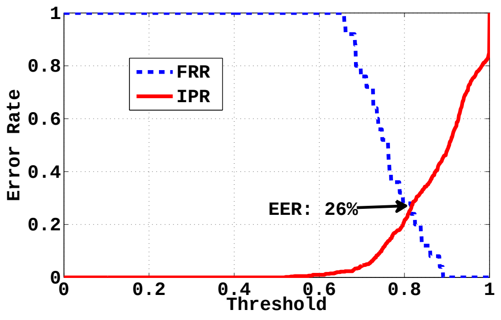
[8] C. Shen, Z. Cai, X. Guan, H. Sha and J. Du, 'Feature Analysis of Mouse Dynamics in Identity Authentication and Monitoring', 2009 IEEE International Conference on Communications, 2009.

[9] A. Syukri, E. Okamoto and M. Mambo, 'A user identification system using signature written with mouse', Information Security and Privacy, pp. 403-414, 1998.

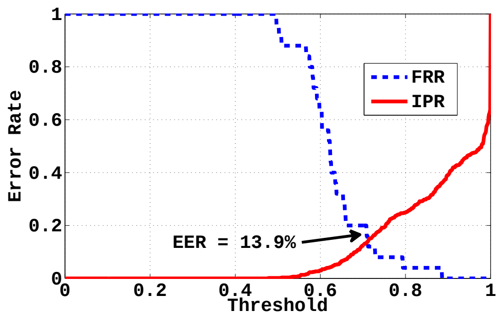
[10] N. Zheng, A. Paloski and H. Wang, 'An efficient user verification system via mouse movements', Proceedings of the 18th ACM conference on Computer and communications security - CCS '11, pp. 139-150, 2011.



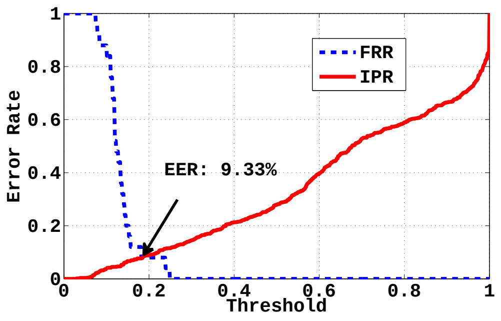
(a) Optimal Settings
Single Click R = 20ms Double Click R = 185ms
M-Value = 2.2



(b) Single Click R = 35ms Double Click R = 195ms
M-Value = 0.7

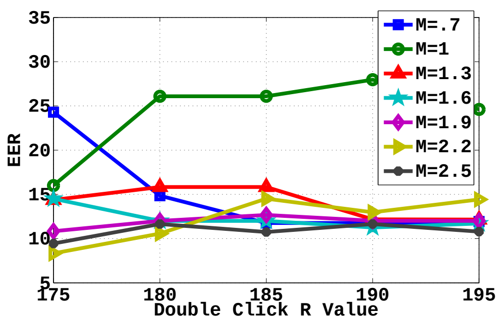


(c) Single Click R = 20ms Double Click R = 180ms
M-Value = 1

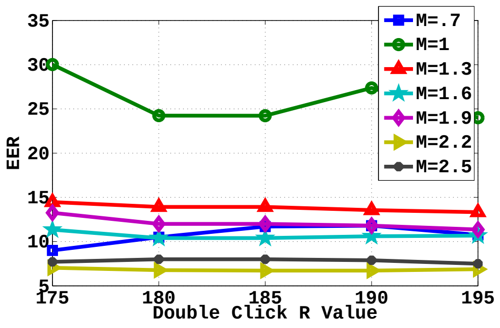


(d) Single Click R = 30ms Double Click R = 190m
M-Value = 1.9

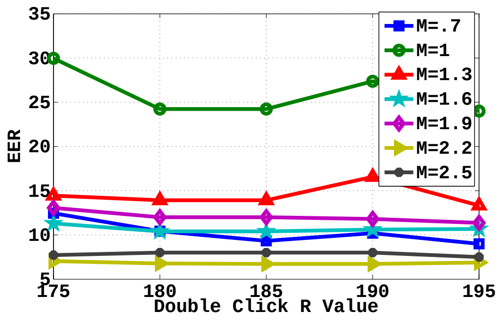
Fig. 3: Plots of the False Rejection Rate and the False Acceptance Rate for four different parameter settings. The point at which the FRR and FAR intersect is the Equal Error Rate.



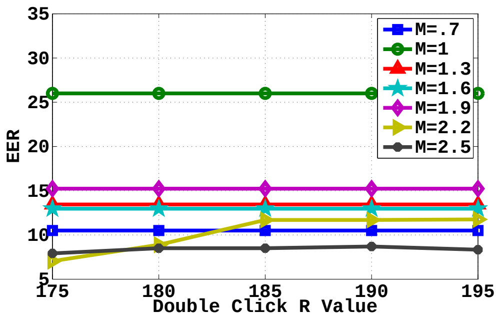
(a) Single Click R = 15ms
Local Max = 24.29% Local Min = 8.33%
Local Average = 14.34% Local STD = 5.08%



(b) Single Click R = 25ms
Local Max = 30.02% Local Min = 6.77%
Local Average = 12.57% Local STD = 6.07%



(c) Single Click R = 30ms
Local Max = 29.96% Local Min = 6.72%
Local Average = 12.58% Local STD = 6.13%



(d) Single Click R = 35ms
Local Max = 26% Local Min = 7.04%
Local Average = 13.82% Local STD = 5.54%

Fig. 4: Scatter plots of EER over the ranges of the parameter values we chose. The X-axis is Double Click R-Values, the Y-axis is the EER. Each plot is a separate Single Click R value.
Global Max = 30.02% Global Min = 6.72% Global Average = 13.19% Global STD = 5.77%