Predictive Statistical Analysis Of Social Media Data

Nicholas Bojanowski
#23947054

Collaborating Group Member: Rudi Plesch

Project Supervisor: Professor Vikram Krishnamurthy

Additional Supervisor: Mustafa Fanaswala

ABSTRACT

The project gathers Twitter volume data and applies statistical predictive models experimentally. Models tested include Markov chains, Markov regime switching, and an order 10 ARMA model, and the testing is done using Matlab packages for TPM and residual analysis. Each subsequent model tested proves more capable of modeling keyword specific Tweet data. The ARMA model is the final model tested, and it uses Gaussian noise. Future models should include exponential noise for more accurate modeling of Tweet volume data. Future experimentation should also include correlation between related keyword datasets and real world events.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

This project experiments with predictive statistical analysis of social media data. It involves the gathering and analysis of social media data in order to develop and test predictive, statistical models, and to explore correlation between social media communication and real world events. Social media usage is extremely prevalent, and an enormous amount of data is being generated. There is a need to understand and model how this massive amount of communication traffic affects, and is affected by, real world events. This project presents a basic building block in that process.

This project is structured experimentally. Methods for gathering data are tried and discarded until an acceptable source can be acquired. Models are proposed, modeled on gathered data, and tested for fit. The results of each model test are used to inform the proposal of the next model, which is again tested for fit. In this way statistical models can be narrowed down and refined to learn a better way of modeling and analyzing the gathered data.

The first objective of this project is to find a source of Twitter volume data that is robust enough for analysis and doesn't come with substantial cost. The second objective is to propose, apply, and analyze the ability of statistical models to properly fit the gathered data. These objectives are accomplished in parallel by student Rudi Plesch with different sets of data and methods of analysis, under the supervision of Professor Vikram Krishnamurthy and his PhD student Mustafa Fanaswala. The original idea for comparing the autocorrelation of residuals between models was proposed by Rudi Plesch to the supervisors.

The following report outlines the steps involved in gathering a source of Twitter data, and the correlation of events in the data with real world events. Once data is gathered, three statistical models are applied to the data, one after the other, each model building on experience from the previous one. The models are a 2-state Markov chain, 2-State Markov Regime Switching, and an autoregressive-moving-average (ARMA) model. Transitional probability matrices and residuals are gathered from each and analyzed to determine model suitability.

This report begins with the project methodology which includes data mining and statistical model generation. Following the methods are the results from the data gathering and model application. Finally, the report ends with a conclusion and the appendices which contain code excerpts, bulk data tables, and residual graphs from the tested models.

2.0 METHODOLOGY

2.1 Data Mining

Prior to the initial gathering of data for analysis, Google Trends [1] was used to search for interesting graphs and trends, assuming that Google Trends would share similar volume results with related Twitter data. The results of these searches led to the creation of targeted keywords that would be used to sort gathered data, focusing on popular tech companies like Apple, Sony, Microsoft, Nintendo, Samsung, and Intel. Initially, these keywords involved stock ticker symbols and company names.

Popular topics on Twitter include entertainment and new technology, and 2013 was a big year for video-game console news. As a result, keyword choices were narrowed down to next-generation video-game console related keywords. Later on, more specific keywords were added to existing keywords searches to see how increased specificity affected volume data.

Data sources investigated include Topsy [2], Datasift [3], Stocktwits [4], Infochimps [5], [6], an ASU database [7], and PeopleBrowsr [8]. Topsy, Datasift, and Stocktwits were discarded, as access to their historic data came with a large price tag. Infochimps provided text dumps of stock tweets from March 2006 to November 2009 relating to the global financial crisis, as well general hashtag sorted tweets from the same time period. The volume data from these tweets, after keyword filtering with Python, were compared with stock market trends. The ASU dataset was downloaded and contained roughly 12 million tweets gathered between February 2011 and August 2011 in the form of tweet IDs. Experimentation was done with the Twitter API and Python to determine the viability of analyzing these tweets. See Appendix A for Python scripts.

Finally, PeopleBrowsr was used to gather Tweet volume data for the generation of statistical models. PeopleBrowsr provides access to 1000 days of volume data for keyword specific searches, downloadable in .csv format. The narrowed down, video-game console related, keywords in Table I were used to obtain roughly 365 days of volume data for the following models. In the event that days were missing data, and average was between the two adjacent days.

TABLE I.
KEYWORD SEARCH TERMS

Keywords		**More Specific Keywords**	
Sony	Nintendo	PS4 Launch	PS4 Used Games
Playstation	Wii	Xbox One Launch	Xbox One Used Games
PS4	Wiiu	PS4 Sales	Sony PS4 Console
Microsoft	Ouya	Xbox One Sales	Microsoft Xbox One Console
Xbox	Oculus Rift	PS4 E3	Nintendo Sony Microsoft
Xbox One		Xbox One E3	PS4 Xbox One

2.2 Statistical Models

2.2.1 Two-State Markov Chains

The first model tested on the data was a 2-state Markov chain. The time-series volume data switches between Markov states and, due to the Markov property, the next state depends only on the current state. Markov states were specified in order to analyze the change in volume over time of keyword-specific Tweets, instead of the volume itself. State 0 was defined as a decrease or lack of change in volume between days. State 1 was defined as an increase in volume between days.

Where $v(t)$ represents Tweet volume, and t is measured in days:
State 0: $v(t) \leq v(t-1)$;
State 1: $v(t) > v(t-1)$.

Each keyword-specific dataset was filtered with a Python script to convert volume data into state data. Using a script in Matlab, the transitional probability matrices (TPMs) were computed for the first and second halves of the state data. These matrices represent the probability of a transition from one Markov state to another. As a basic test to see if the probabilities calculated for the first halves could be used to predict the second halves of the data, the difference between the TPMs were calculated for each keyword. See Appendix A for Python and Matlab scripts.

2.2.2 Two-State Markov Regime Switching

The second model tested on the data was a Markov switching model. This model was the next logical step as it involves a first-order Markov chain, bridging the gap between basic Markov chains and the AR models that are commonly used in statistical analysis. This was accomplished using the MG_Regress package [9] in Matlab.

The time-series volume data was input for each keyword, as well each series' differential. Prior to calculation, each set of data was preprocessed and feature scaled. This normalization was done by subtracting the mean of the data from the data, and then dividing by the standard deviation, as seen in Figure 1.

$$data = \frac{\left(data - mean(data)\right)}{std(data)}$$

Fig. 1. Data Normalization

The inputs follow the outline documented in the package documentation [9]. The dependent variable, $y(t)$, was defined as the time series starting at the second entry: y(2:end). The independent variable, $y(t-1)$, was defined as that time series shifted back by one: y(1:end-1). See Appendix A for the general Matlab script.

MS_Regress was run on both the first half and second half of each dataset. The TPM was taken out of each data structure and, as in 2.2.1, the difference was taken between the TPMs of the first and second halves of data.

Finally, MS_Regress was run on the full span of each dataset. The residuals for the model were then extracted. Using methods described in the Matlab documentation [11], the residuals were analyzed and graphed using autocorrelation and 99% confidence intervals.

2.2.3 Autoregressive-Moving-Average (ARMA)

The final model tested on the data was an autoregressive-moving-average model. This model was chosen as it has commonly been used to successfully analyze similar time-series data such as stock market fluctuations or speech. Model generation and testing was accomplished using the "armax" function in Matlab, found in the System Identification Toolbox [10]. Figure 2 shows the formula used by this function, as the modeled data is a time series with a single output channel, and no exogenous input channel.

$$A(q)y(t) = e(t)$$

Fig. 2. ARMA Formula

As described in section 2.2.2, the differential versions of the keyword-specific time-series data was generated, and all sets of data were preprocessed and normalized. The order of the ARMA model was 10, with AR order 5 and MA order 5, as the number of data points in the time series was roughly 365. See Appendix A for the general Matlab script.

Residuals were generated by comparing the generated model with the original data. These residuals were graphed and examined to determine model suitability, like in section 2.2.2.

3.0 RESULTS

3.1 Time-Series Data

Topsy, Datasift, and Stocktwits data proved to be too expensive for the scope of this project, and no data was gathered from them. Infochimps' data was specific to a very particular time period (the global financial crisis), and as it took place early in Twitter's lifespan many keyword-specific searches resulted in relatively low volumes of Tweets. The ASU dataset contained only TweetIDs due to privacy concerns. Getting each full tweet for the entire dataset was deemed infeasible, as experimentation with the Twitter API revealed strict caps on calls to their database for Tweet information. Peoplebrowsr's 1000 day searchable Tweet history proved a valuable source of data for statistical modeling.

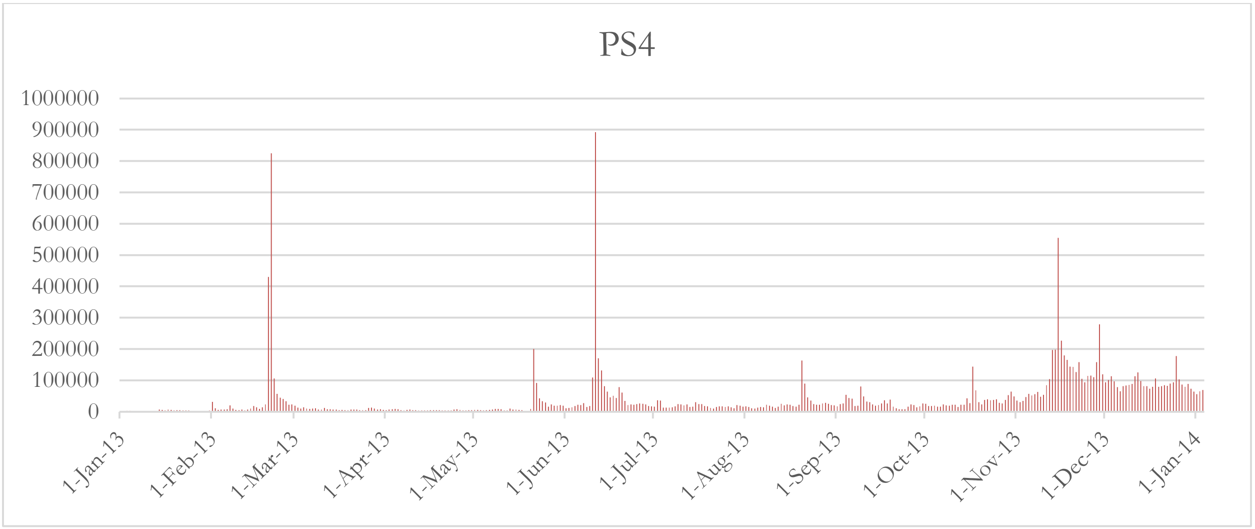

Fig. 3. PS4 Data

The time-series data from the keyword search terms seen in Table I was all similar in nature. Figure 3 shows the graph for the keyword "PS4". Tweet volume is seen to reach extreme peaks at multiple points in the series. As keywords increase in specificity from company name to singular product (Sony -> Playstation -> PS4), the peaks often remain while the overall conversation volume decreases. The lower volumes represent the overall ebb and flow of Twitter conversation, while the spikes in volume are directly related to official corporate announcements, competitor announcements, product release dates, and annual scheduled trade shows, exhibitions, and conferences. Table II shows a list of events related to the spikes in Figure 3.

TABLE II.
PS4 SPIKE EVENTS

Feb. 20, 2013	PS4 Announced at SCE Press Conference	**Aug. 20, 2013**	Gamescon 2013
Feb. 21, 2013	More info given at same Press Conference	**Sept. 9, 2013**	Sony Press Conference
May 21, 2013	Competitor Xbox One Announced	**Oct. 17, 2013**	Brazilian price of US $1759 announced
June 11, 2013	E3 2013	**Nov. 15, 2013**	PS4 Release Date
June 19, 2013	Competitor Xbox DRM Cuts Announced	**Nov. 29, 2013**	PS4 Released in Europe

Adding additional keywords to increase specificity decreased overall Tweet volume to the point where many days on the time series had 0 keyword mentions at all. Due to this, only the original specific keywords on the left side of Table I were used in the statistical models.

See Appendix B for more graphs of time-series data.

3.2 Statistical Models

3.2.1 Two-State Markov Chains

Most of the 2-state Markov chain transitional probability matrixes for the data were split around 50/50 to 60/40, as seen in Table III. This implies that even though the Markov state is switching constantly throughout time-series, it's doing so evenly.

TABLE III.
PS4 TPM

TPM: First Half of Data				TPM: Second Half of Data		
PS4	**Down – 0**	**Up – 1**		**PS4**	**Down – 0**	**Up – 1**
Down – 0	0.6078	0.3922		**Down – 0**	0.5638	0.4362
Up – 1	0.5125	0.4875		**Up – 1**	0.4659	0.5341

The important part of this model's tests involved the differences between the TPM of each half of the data, for each time-series. At worst, the difference between halves was less than 12%, as seen in Table IV. At best, the difference was less than 2%, as seen in Table V. This suggests that the Markov model applied to the first half of the data could be capable of predicting the data in the second half, making further model generation worthwhile.

TABLE IV.
OCULUS RIFT TPM DIFFERENCE

Oculus Rift	Down – 0	Up – 1
Down – 0	-0.1143	0.1143
Up – 1	0.0301	-0.0301

TABLE V.
WII TPM DIFFERENCE

Wii	Down – 0	Up – 1
Down – 0	-0.0049	0.0049
Up – 1	0.0196	-0.0196

The keywords with the smallest difference are “Wii”, “Nintendo”, and “PS4”, indicating that those datasets are providing the best fits. See Appendix C for additional calculated TPMs.

3.2.2 Two-State Markov Regime Switching

The differences between the TPMs of the first and second halves of data modeled by the MS_Regress package varied between datasets. Some datasets resulted in small differences similar to the results of 3.2.1. Table VI shows an example of this. Other datasets, like the example in Table VII, demonstrated TPMs differences that were drastically different.

TABLE VI.
MS_REGRESS WII TPM DIFFERENCES

Wii	**0**	**1**
0	-0.0535	-0.0928
1	0.0535	0.0928

dWii	**0**	**1**
0	-0.0564	-0.0128
1	0.0564	0.0128

TABLE VII.
MS_REGRESS PS4 TPM DIFFERENCES

PS4	**0**	**1**
0	-0.2312	-0.5605
1	0.2312	0.5605

dPS4	**0**	**1**
0	-0.0281	0.0774
1	0.0281	-0.0774

This indicates that, as expected, the states generated by MS_Regress are quite different from the established states of the previous model. This also indicates that simply comparing TPMs for first and second halves with the MS_Regress package does not work for every dataset, and is not particularly effective at demonstrating the fit quality of the models. See Appendix D for additional TPMs.

The autocorrelation graphs of the residuals show that for the undifferentiated time-series datasets, the majority of the values lie within the 99% confidence bounds as seen in Figure 4. The differentiated data sets contain more values that are outside the bounds, particularly around lag -2 to 2, as seen in Figure 5.

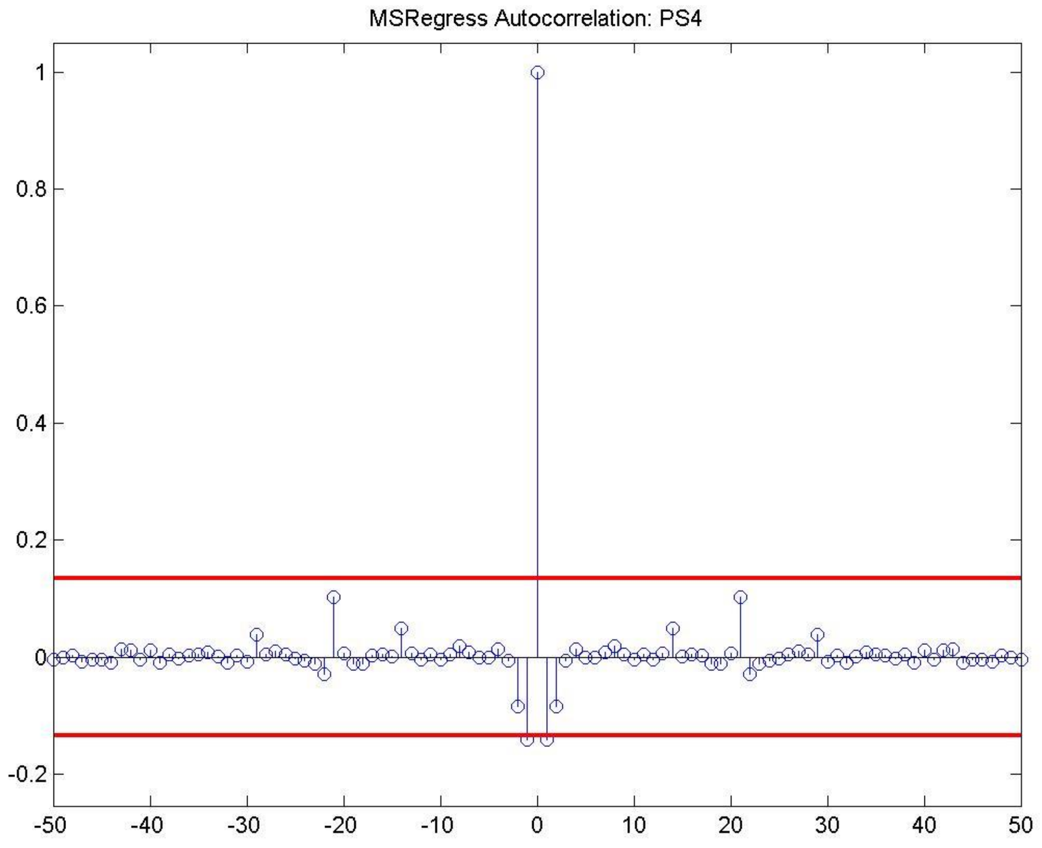

Fig. 4. MSRegress PS4 Residuals

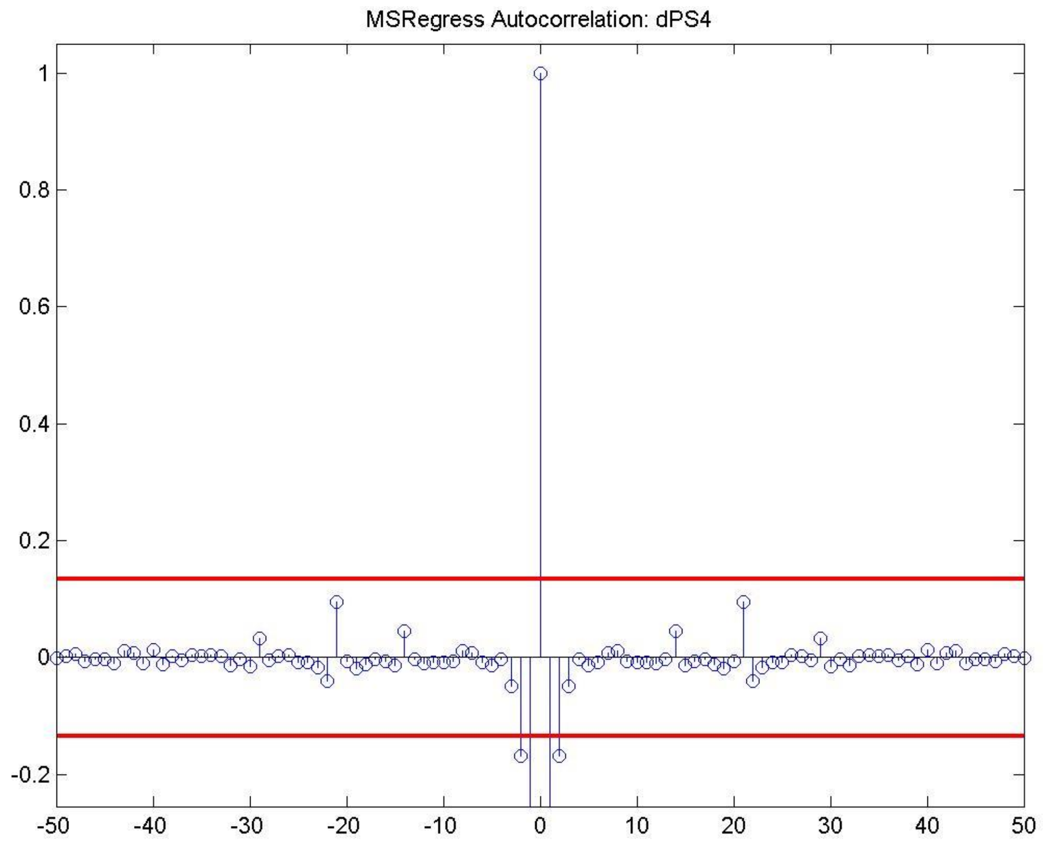

Fig. 5. MSRegress dPS4 Residuals

This implies that the models are accounting for most, if not all, of the signals in the undifferentiated time-series datasets, and that the residuals consist of noise. This holds for the differentiated datasets as well, but to a lesser extent. See Appendix E for MS_Regress residual analysis graphs.

3.2.3 Autoregressive Moving-Average (ARMA)

The Autocorrelated residuals from the ARMA model demonstrated a large improvement over previous model. Figure 6 shows the autocorrelated residual graph for the “PS4” keyword, and Figure 7 shows the autocorrelated residual graph for its derivative, “dPS4”.

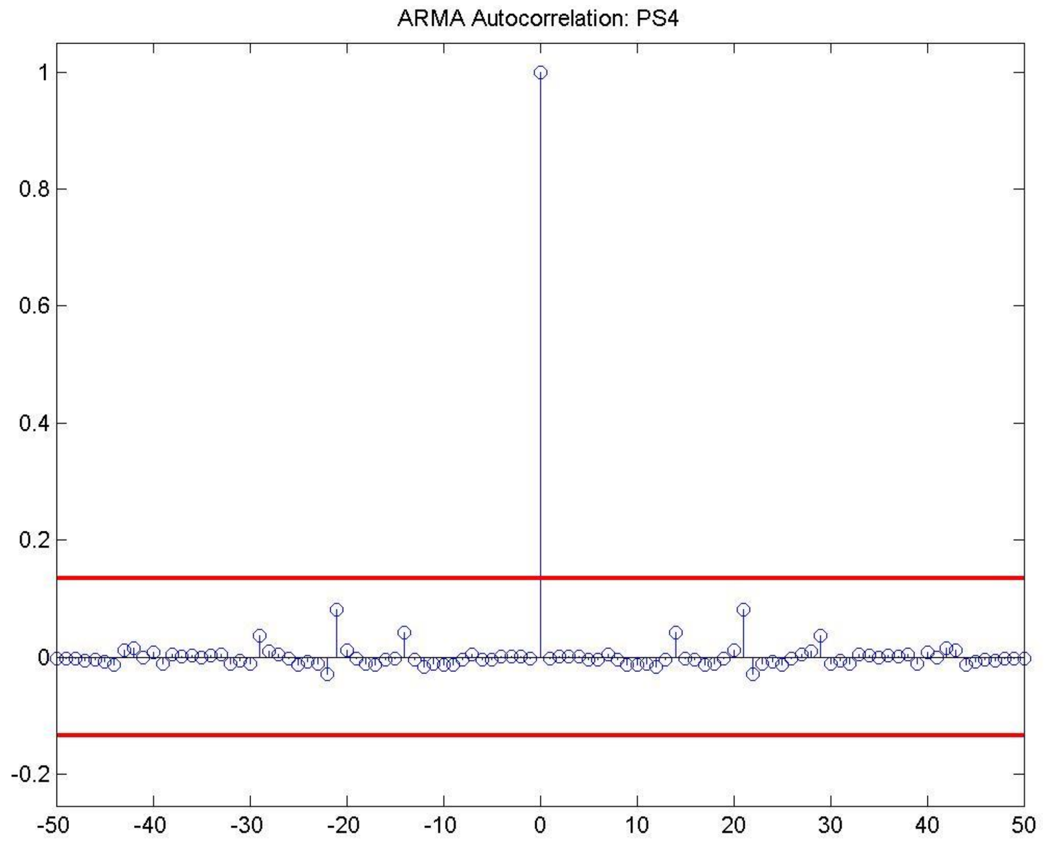

Fig. 6. ARMA PS4 Residuals

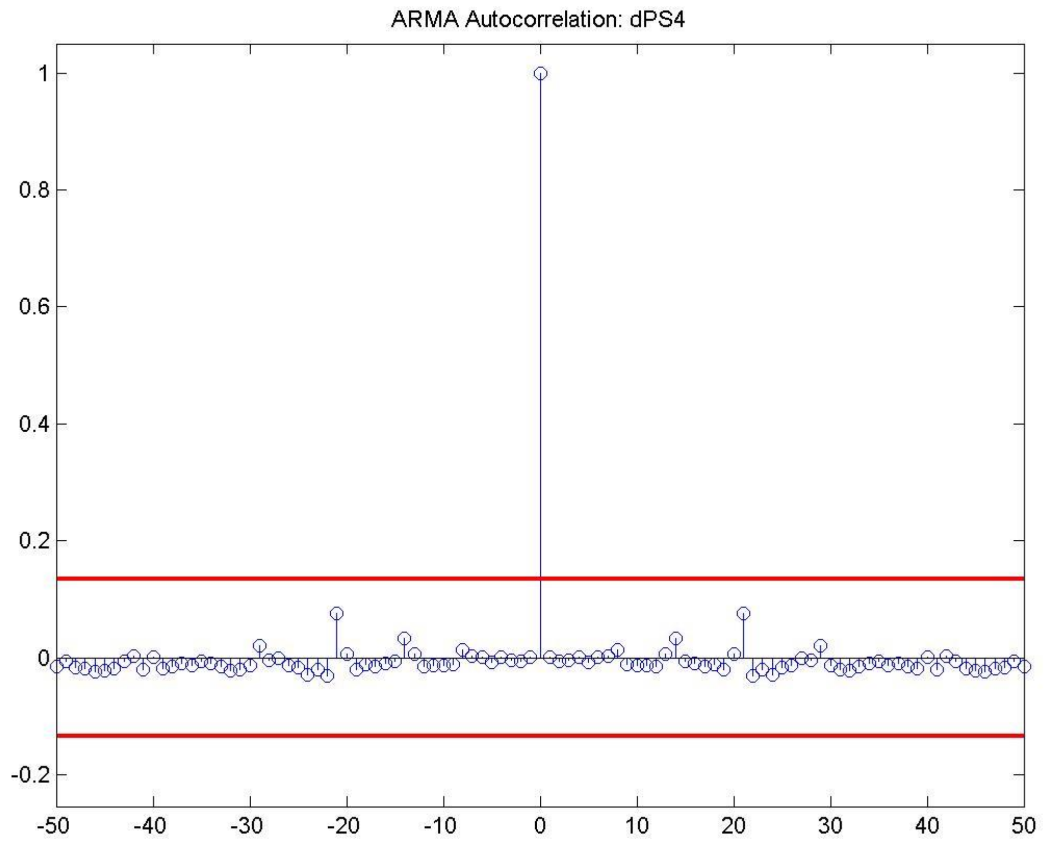

Fig. 7. ARMA dPS4 Residuals

With the exception of a few datasets, the majority of modeled data produced values completely bound by the 99% confidence intervals. The differentiated datasets behaved the same way, in contrast with results in 3.2.2. This indicates that the ARMA model does a better job at accounting for accounting for all of the signals, and that the residuals contain mostly white noise. See Appendix F for ARMA residual analysis graphs.

3.3 Results Summary

Each subsequent statistical model applied to the datasets was more capable of modeling and predicting the data than the model before. There is no perfect model, but this process of refinement can proceed into the future with the testing of more complicated statistical models. The next step, specifically, would involve the use of ARMA modeling with exponential noise, as the time-series data contains exponential noise and the models used in this project rely on Gaussian noise. The goals of the project were achieved, but there is always room for improvement and further model generation.

The objectives achieved include exploring sources of Twitter volume data, and the application and testing of predictive, statistical models to gathered time-series datasets. Basic event to data correlation was done, but the project did not progress to the point of statistically analyzing datasets with related keywords for cross-correlation.

Modeling and prediction of overall Tweet volume appears feasible, but it's likely quite difficult to predict the enormous spikes of volume that occur when product announcements are made based on the data alone, without access to outside sources of information. While the large events connected to the volume spikes may be difficult to predict, monitoring Twitter streams for enormous spikes and running analysis on keywords in the spiking conversation may be an effective way of pinpointing and localizing events as/after they occur.

4.0 CONCLUSION

This report experimented with predictive statistical models and their application on social media data. Twitter time-series volume data was gather from PeopleBrowsr. As Twitter conversation often involves entertainment and technology, search keywords were chose to reflect their combination with recent video-game console technologies.

The first model tested on the data was a 2-state Markov chain, and the difference in TPMs for the datasets suggested that a model applied to the first half of the data could be used to predict the second half. The second model tested was the MS_Regress implementation of Markov regime switching model. The TPMs of this model behaved differently from the original Markov chain, and prompted analysis of the autocorrelation of residuals. The final model tested as an order 10 ARMA model. The residuals produced by the application of this model almost all fell within the bounds of 99% confidence, implying that the residuals of the model included no signal information and only white noise.

Future model testing should begin with ARMA models with exponential noise, as this would be an immediate improvement on the Gaussian noise models used. The prevalence of enormous volume spikes may be difficult to predict with statistical models. Statistical modeling could provide a baseline from which spikes could be detected in Twitter streams, which could provide future analysis of Tweet content in the spike for event localization and specification.

REFERENCES

[1] Google. "Google Trends". [Online]. Internet: www.google.ca/trends/ [Jan. 17, 2014]

[2] Topsy Labs, Inc. "Topsy". [Online]. Internet: www.topsy.com [Jan. 23, 2014]

[3] Datasift. "Datasift". [Online]. Internet: www.datasift.com [Jan. 23, 2014]

[4] StockTwits. "StockTwits". [Online]. Internet: www.stocktwits.com [Jan. 23, 2014]

[5] Monkeywrench Consultancy. (2010). "Twitter Census: Stock Tweets". [Online]. Internet: http://www.infochimps.com/datasets/twitter-census-stock-tweets [Jan 23, 2014]

[6] Monkeywrench Consultancy. (2010). "Twitter Census: Hashtags, URLS, Smileys by Day". [Online]. Internet: http://www.infochimps.com/datasets/twitter-census-hashtags-urls-smileys-by-day [Jan 23, 2014]

[7] X. Wang, H. Liu, P. Zhang, B. Li. (2012). "Identifying Information Spreaders in Twitter Follower Networks". [Online]. Internet: http://dmml.asu.edu/users/xufei/datasets.html [Jan. 30, 2014]

[8] PeopleBrowsr. "PeopleBrowsr Analytics". [Online]. Internet: www.analytics.peoplebrowsr.com [Feb. 23, 2014]

[9] M. Perlin. "MS_Regress – A Package for Markov Regime Switching Models in Matlab". [Online]. Internet: https://sites.google.com/site/marceloperlin/matlab-code/ms_regress---a-package-for-markov-regime-switching-models-in-matlab [April. 4, 2014]

[10] Matlab. "Estimating AR and ARMA Models". [Online]. Internet: http://www.mathworks.com/help/ident/ug/estimating-ar-and-arma-models.html [April. 4, 2014]

[11] Matlab. "Residual Analysis with Autocorrelation". [Online]. Internet: http://www.mathworks.com/help/signal/ug/residual-analysis-with-autocorrelation.html [April. 4, 2014]

APPENDIX A: Python and Matlab Scripts

TweetID to Tweet (Python)

```
import twitter
import re
import sys

reload(sys);
sys.setdefaultencoding("utf8")

dataset = open('./TweetIdTest.txt')
outFile = open('./fullTweets.txt', 'a')

consumerKey = '#####'
consumerSecret = '#####'
accessTokenKey = '#####'
accessTokenSecret = '#####'

api = twitter.Api(consumerKey, consumerSecret, accessTokenKey,
accessTokenSecret)

for line in dataset:
     tweet = api.GetStatus(line, include_entities=True)
     outString = line

     for i in tweet.hashtags:
          outString += ('#' + i.text)

     outString += tweet.text
     outFile.write(outString + '\n')

dataset.close()
```

Markov States from Volume (Python)

```
import sys

dataset = [line.strip() for line in open('./data.txt')]

outFile = open('./states.txt', 'a')

for i in range(0,300):
     if (int(dataset[i]) < int(dataset[i+1])):
          outFile.write('1' + '\n')
     else:
          outFile.write('0' + '\n')

outFile.close()
```

TPM from Data (Matlab)

```
count = full(sparse(x(1:end-1), x(2:end), 1))

matrix = bsxfun(@rdivide, count, sum(count, 2))
```

MS REGRESS (Matlab)

```
dep = (data(2:end));
indep = (data(1:end-1);

S = [1 1]; % starting state
k = 2; % number of states
advOpt.distrib = 'Normal';
advOpt.std_method = 1;
advOpt.doPlots = 0;

output = MS_Regress_Fit(dep, indep, k, S, advOpt);

residuals = out.resid; %Extracts Residuals
```

ARMAX System ID Toolbox (Matlab)

```
AR = 5; % Autoregressive Order
MA = 5; % Moving-Average Order

output = armax(data, [AR MA]);

residGraph = figure, e = resid(output,data);

residuals = resid(out, inData); %Extracts Residuals
```

Autocorrelation, 99% Confidence Interval, Residual Graphing (Matlab)

```matlab
% Print Residuals, 99% Confidence Intervals / Autocorrelation
[xc lags] = xcorr(residuals, 50, 'coeff');

conf99 = sqrt(2)*erfcinv(2*0.01/2);
lconf = -conf99/sqrt(length(residuals));
upconf = conf99/sqrt(length(residuals));

plot = figure, stem(lags,xc);

set(gca,'ylim',[lconf-0.12 1.05]);
hold on;
line(lags,lconf*ones(size(lags)),'color','r','linewidth',2);
line(lags,upconf*ones(size(lags)),'color','r','linewidth',2);

titlename = ['MSRegress/ARMA Autocorrelation: ' fileName];
title(titlename);

outname = ['MsRegress/ARMA' fileName '.jpg']

saveas(plot,outname)
```

APPENDIX B: Time-Series Graphs

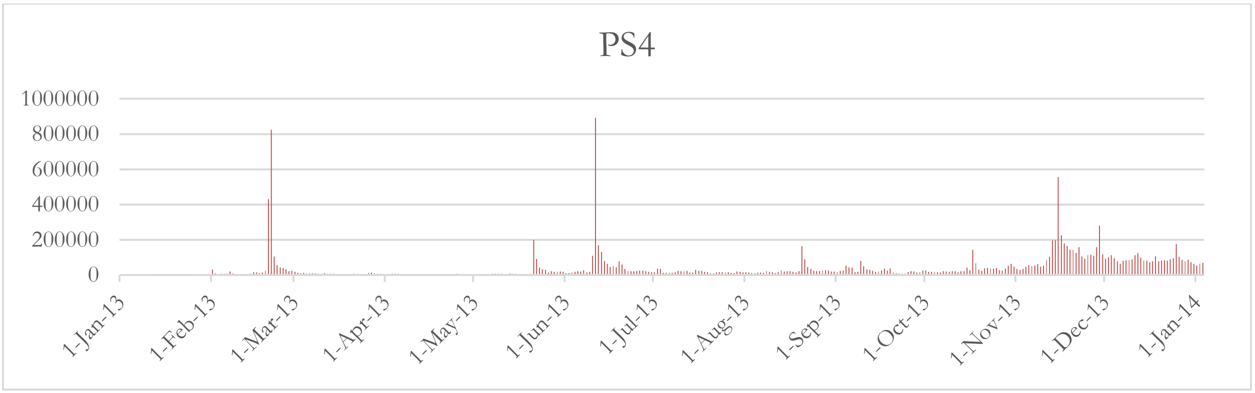

Fig. 8. PS4 Data

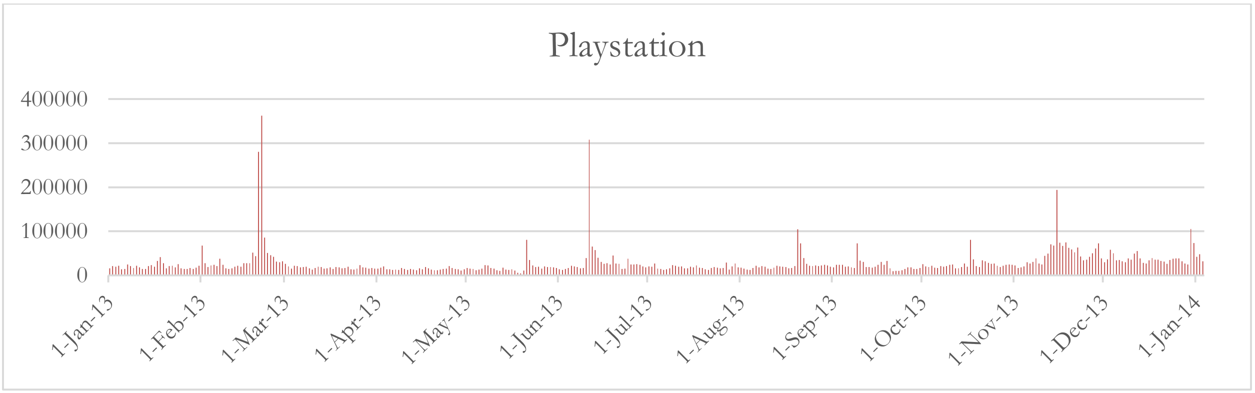

Fig. 9. Playstation Data

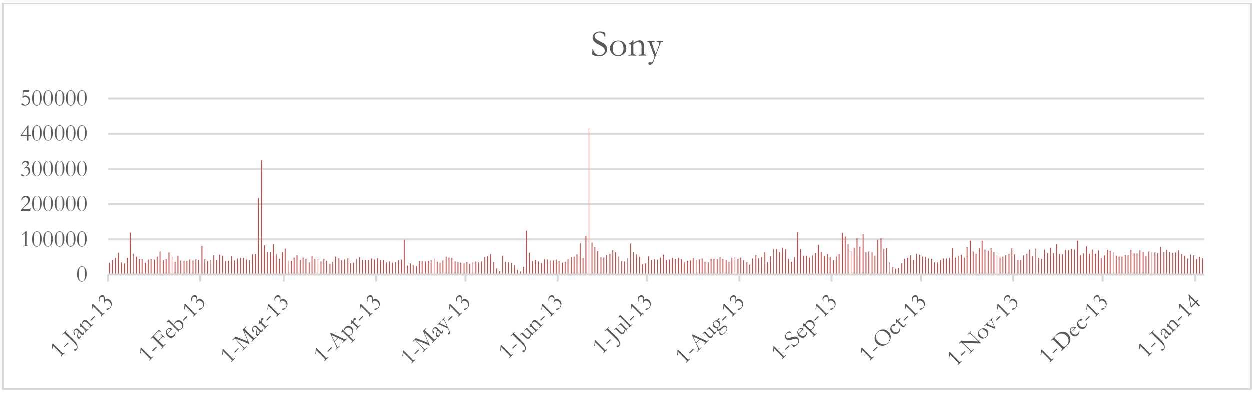

Fig. 10. Sony Data

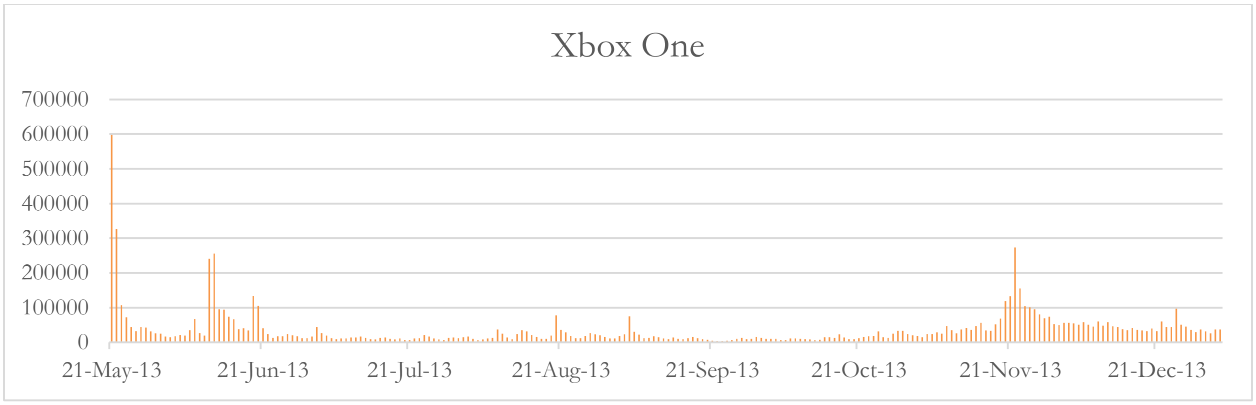

Fig. 11. Xbox One Data

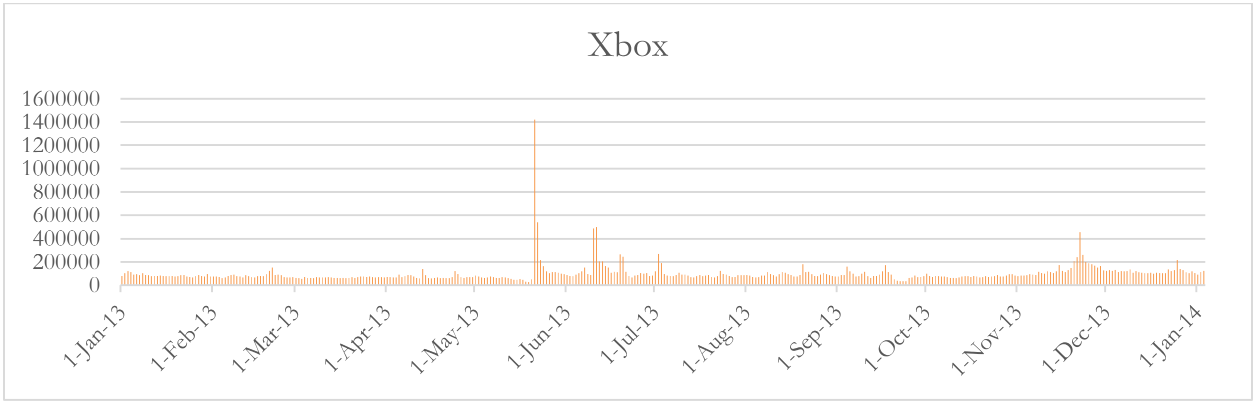

Fig. 12. Xbox Data

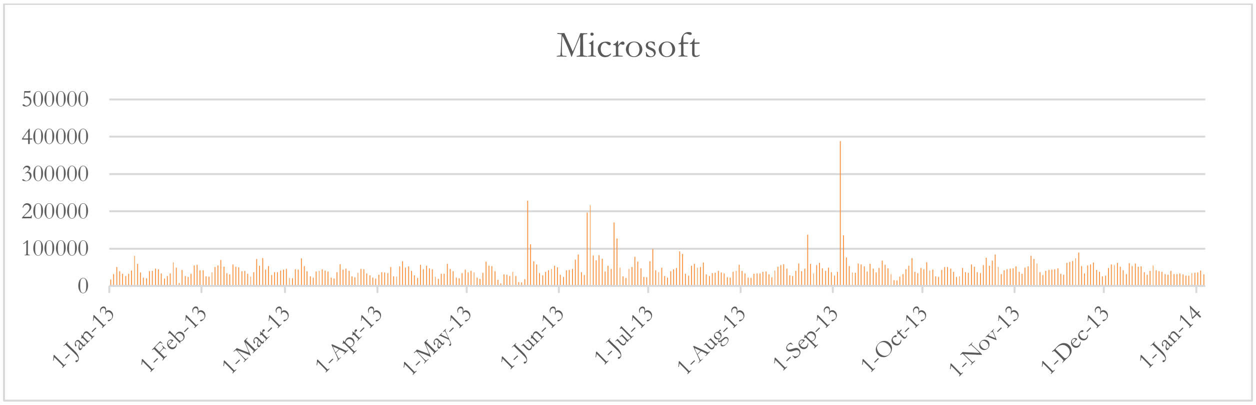

Fig. 13. Microsoft Data

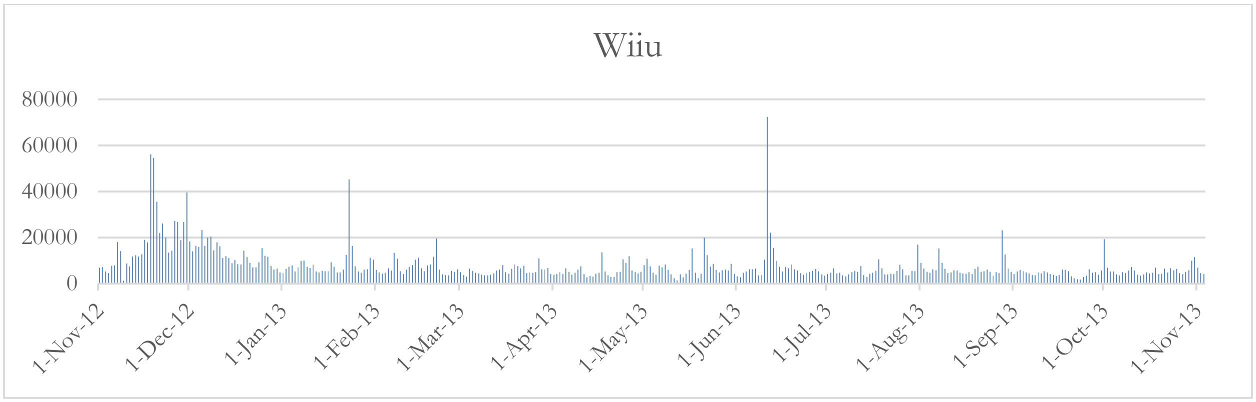

Fig. 14. Wiiu Data

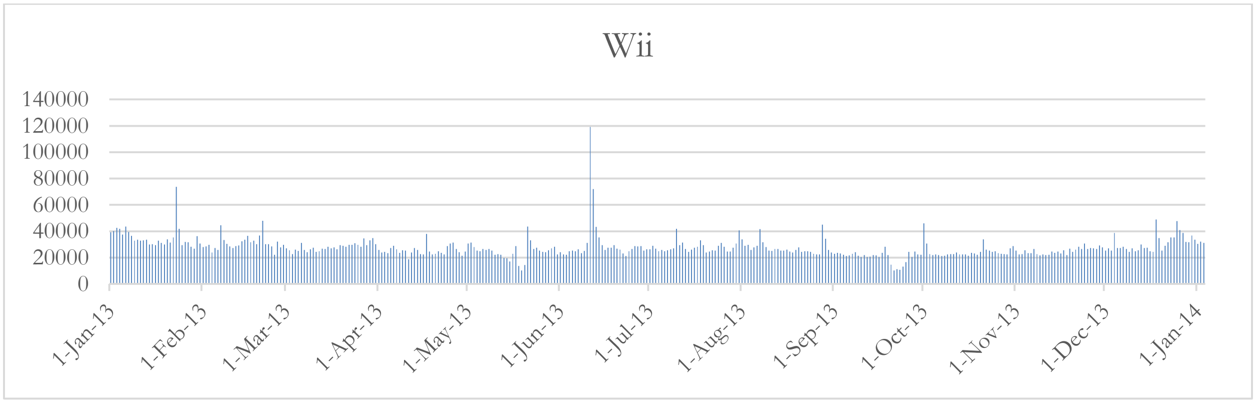

Fig. 15. Wii Data

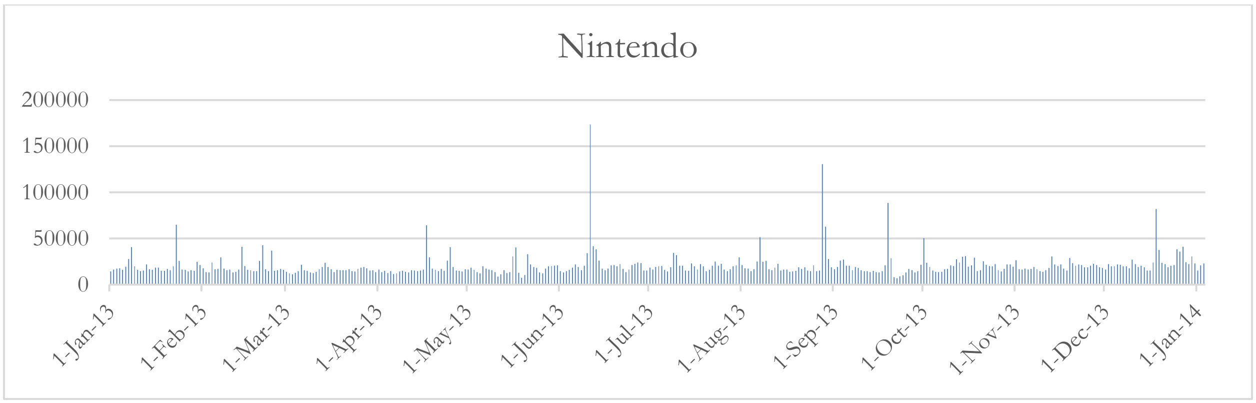

Fig. 16. Nintendo Data

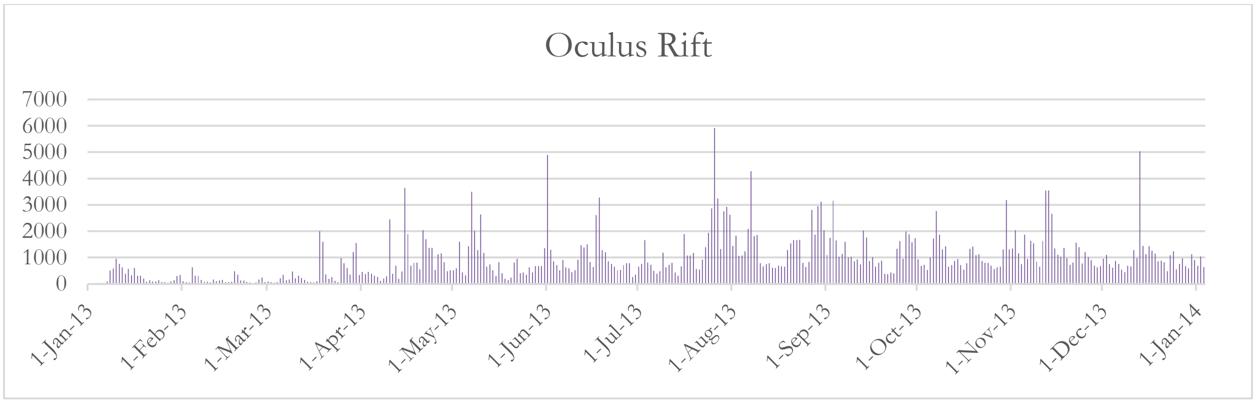

Fig. 17. Oculus Rift Data

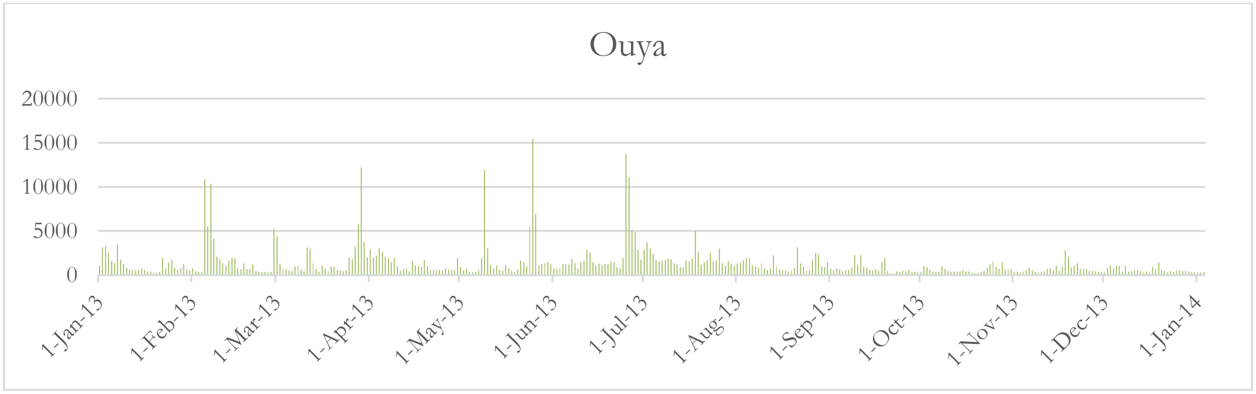

Fig. 18. Ouya Data

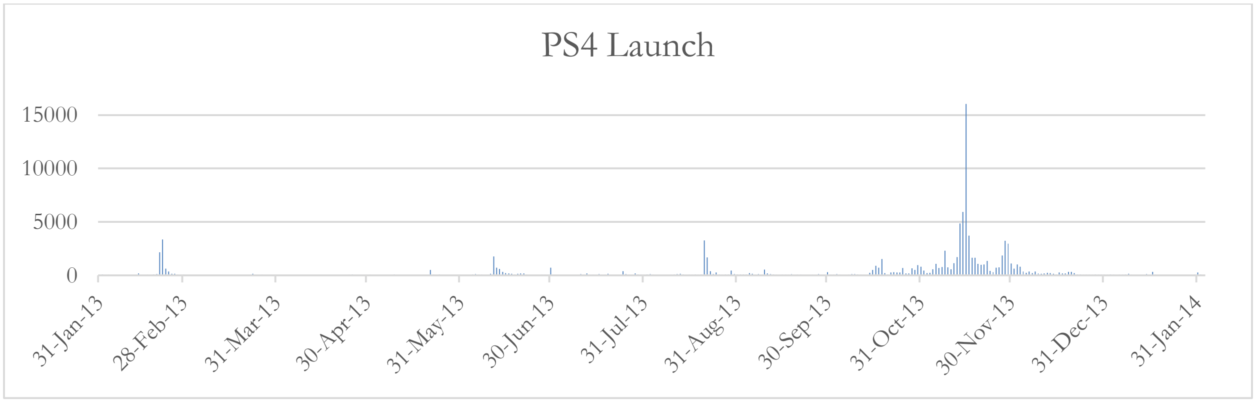

Fig. 19. PS4 Launch Data

APPENDIX C: Markov TPMs

TABLE VIII.
MARKOV CHAIN TPMS

TPM: First Half of Data				TPM: Second Half of Data		
PS4	**Down – 0**	**Up – 1**		**PS4**	**Down – 0**	**Up – 1**
Down – 0	0.6078	0.3922		**Down – 0**	0.5638	0.4362
Up – 1	0.5125	0.4875		**Up – 1**	0.4659	0.5341
Playstation	**Down – 0**	**Up – 1**		**Playstation**	**Down – 0**	**Up – 1**
Down – 0	0.5437	0.4563		**Down – 0**	0.5408	0.4592
Up – 1	0.5949	0.4051		**Up – 1**	0.5357	0.4643
Sony	**Down – 0**	**Up – 1**		**Sony**	**Down – 0**	**Up – 1**
Down – 0	0.5213	0.4787		**Down – 0**	0.4409	0.5591
Up – 1	0.5114	0.4886		**Up – 1**	0.5955	0.4045
Xbox One	**Down – 0**	**Up – 1**		**Xbox One**	**Down – 0**	**Up – 1**
Down – 0	0.6970	0.3030		**Down – 0**	0.5873	0.4127
Up – 1	0.4348	0.5652		**Up – 1**	0.5306	0.4694
Xbox	**Down – 0**	**Up – 1**		**Xbox**	**Down – 0**	**Up – 1**
Down – 0	0.5810	0.4190		**Down – 0**	0.5800	0.4200
Up – 1	0.5844	0.4156		**Up – 1**	0.5000	0.5000
Microsoft	**Down – 0**	**Up – 1**		**Microsoft**	**Down – 0**	**Up – 1**
Down – 0	0.6154	0.3846		**Down – 0**	0.5714	0.4286
Up – 1	0.5256	0.4744		**Up – 1**	0.4396	0.5604
Wiiu	**Down – 0**	**Up – 1**		**Wiiu**	**Down – 0**	**Up – 1**
Down – 0	0.5213	0.4787		**Down – 0**	0.5446	0.4554
Up – 1	0.5227	0.4773		**Up – 1**	0.5679	0.4321
Wii	**Down – 0**	**Up – 1**		**Wii**	**Down – 0**	**Up – 1**
Down – 0	0.4796	0.5204		**Down – 0**	0.4747	0.5253
Up – 1	0.6190	0.3810		**Up – 1**	0.6386	0.3614
Nintendo	**Down – 0**	**Up – 1**		**Nintendo**	**Down – 0**	**Up – 1**
Down – 0	0.5521	0.4479		**Down – 0**	0.5361	0.4639
Up – 1	0.5000	0.5000		**Up – 1**	0.5294	0.4706

Continued on Next Page…

TPM: First Half of Data

Ouya	Down – 0	Up – 1
Down – 0	0.6195	0.3805
Up – 1	0.6377	0.3623
Oculus Rift	**Down – 0**	**Up – 1**
Down – 0	0.5600	0.4400
Up – 1	0.5366	0.4634

TPM: Second Half of Data

Ouya	Down – 0	Up – 1
Down – 0	0.5102	0.4898
Up – 1	0.5595	0.4405
Oculus Rift	**Down – 0**	**Up – 1**
Down – 0	0.4457	0.5543
Up – 1	0.5667	0.4333

TABLE IX.
MARKOV CHAIN TPM DIFFERENCES

PS4	Down – 0	Up – 1
Down – 0	-0.0440	0.0440
Up – 1	-0.0466	0.0466
Playstation	**Down – 0**	**Up – 1**
Down – 0	-0.0029	0.0029
Up – 1	-0.0592	0.0592
Sony	**Down – 0**	**Up – 1**
Down – 0	-0.0804	0.0804
Up – 1	0.0841	-0.0841
Wiiu	**Down – 0**	**Up – 1**
Down – 0	0.0233	-0.0233
Up – 1	0.0452	-0.0452
Wii	**Down – 0**	**Up – 1**
Down – 0	-0.0049	0.0049
Up – 1	0.0196	-0.0196
Nintendo	**Down – 0**	**Up – 1**
Down – 0	-0.0160	0.0160
Up – 1	0.0294	-0.0294

Xbox One	Down – 0	Up – 1
Down – 0	-0.1097	0.1097
Up – 1	0.0958	-0.0958
Xbox	**Down – 0**	**Up – 1**
Down – 0	-0.1000	0.1000
Up – 1	-0.0844	0.0844
Microsoft	**Down – 0**	**Up – 1**
Down – 0	-0.0440	0.0440
Up – 1	-0.0860	0.0860
Oculus Rift	**Down – 0**	**Up – 1**
Down – 0	-0.1143	0.1143
Up – 1	0.0301	-0.0301
Ouya	**Down – 0**	**Up – 1**
Down – 0	-0.1093	0.1093
Up – 1	-0.0782	0.0782

APPENDIX D: MS Regress TPMs

TABLE X.
MS_REGRESS TPMS

TPM: First Half of Data		
PS4	**0**	**1**
0	0.9031	0.6179
1	0.0969	0.3821
dPS4	**0**	**1**
0	0.9580	0.3028
1	0.0420	0.6972
Xbox	**0**	**1**
0	0.9437	0.3000
1	0.0563	0.7000
dXbox	**0**	**1**
0	0.9467	0.3376
1	0.0533	0.6624
Wii	**0**	**1**
0	0.9507	0.4384
1	0.0493	0.5616
dWii	**0**	**1**
0	0.9518	0.3281
1	0.0482	0.6719
Nintendo	**0**	**1**
0	0.9164	0.3019
1	0.0836	0.6981
dNintendo	**0**	**1**
0	0.9126	0.3370
1	0.0874	0.6630

TPM: Second Half of Data		
PS4	**0**	**1**
0	0.6719	0.0574
1	0.3281	0.9426
dPS4	**0**	**1**
0	0.9299	0.3801
1	0.0701	0.6199
Xbox	**0**	**1**
0	0.9494	0.3131
1	0.0506	0.6869
dXbox	**0**	**1**
0	0.9371	0.3674
1	0.0629	0.6326
Wii	**0**	**1**
0	0.8972	0.3456
1	0.1028	0.6544
dWii	**0**	**1**
0	0.8954	0.3153
1	0.1046	0.6847
Nintendo	**0**	**1**
0	0.9536	0.6005
1	0.0464	0.3995
dNintendo	**0**	**1**
0	0.9415	0.4510
1	0.0585	0.5490

TABLE XI.
MS_REGRESS TPM DIFFERENCES

PS4	**0**	**1**
0	-0.2312	-0.5605
1	0.2312	0.5605
Xbox	**0**	**1**
0	0.0057	0.0131
1	-0.0057	-0.0131
Wii	**0**	**1**
0	-0.0535	-0.0928
1	0.0535	0.0928
Nintendo	**0**	**1**
0	0.0371	0.2986
1	-0.0371	-0.2986

dPS4	**0**	**1**
0	-0.0281	0.0774
1	0.0281	-0.0774
dXbox	**0**	**1**
0	-0.0097	0.0298
1	0.0097	-0.0298
dWii	**0**	**1**
0	-0.0564	-0.0128
1	0.0564	0.0128
dNintendo	**0**	**1**
0	0.0289	0.1140
1	-0.0289	-0.1140

APPENDIX E: MS Regress Residuals

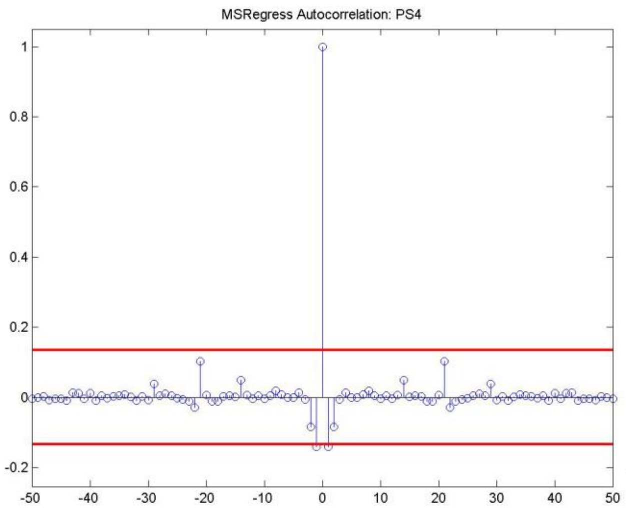

Fig. 20. MSRegress PS4 Residuals

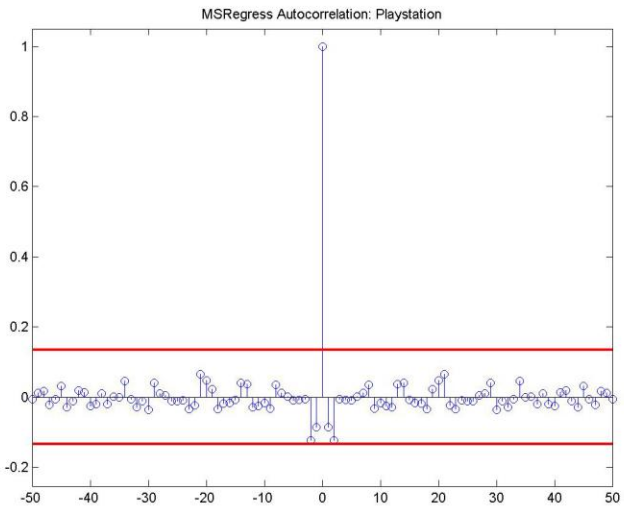

Fig. 21. MSRegress Playstation Residuals

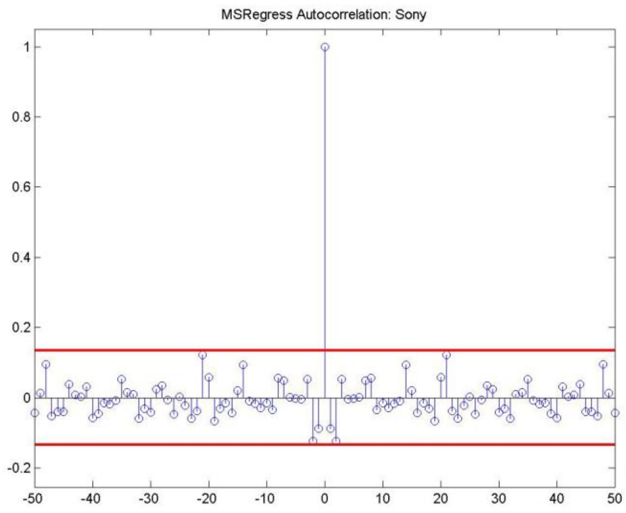

Fig. 22. MSRegress Sony Residuals

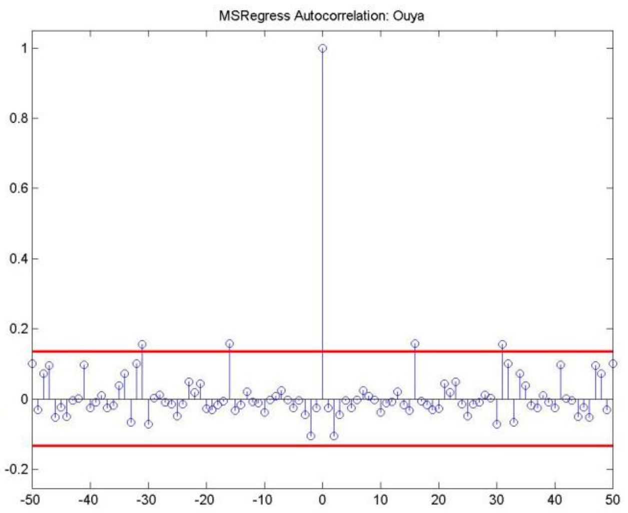

Fig. 23. MSRegress Ouya Residuals

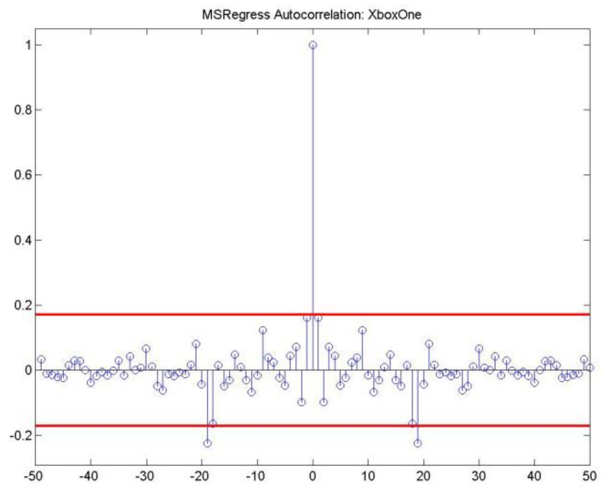

Fig. 24. MSRegress Xbox One Residuals

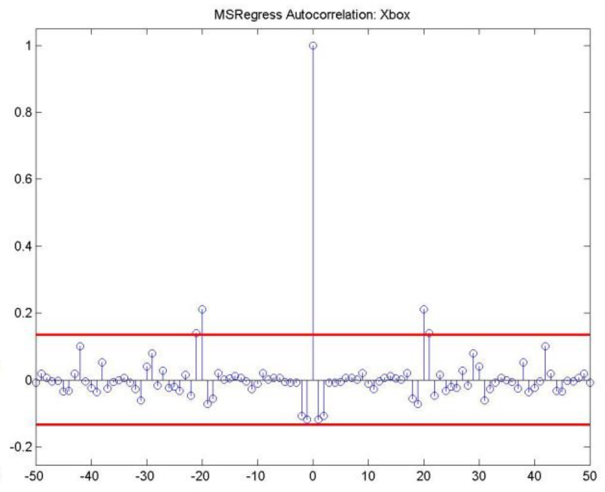

Fig. 25. MSRegress Xbox Residuals

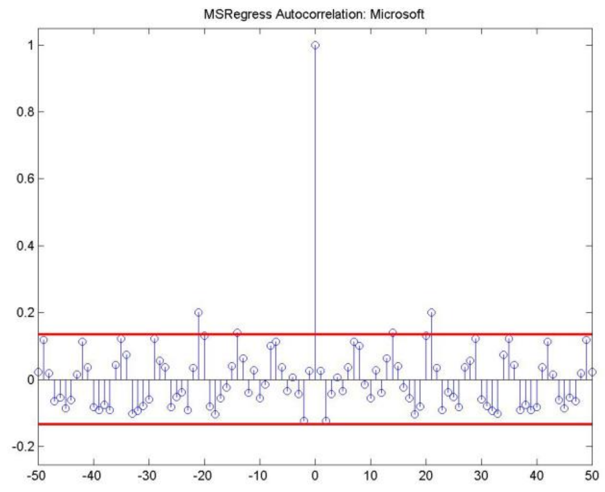

Fig. 26. MSRegress Microsoft Residuals

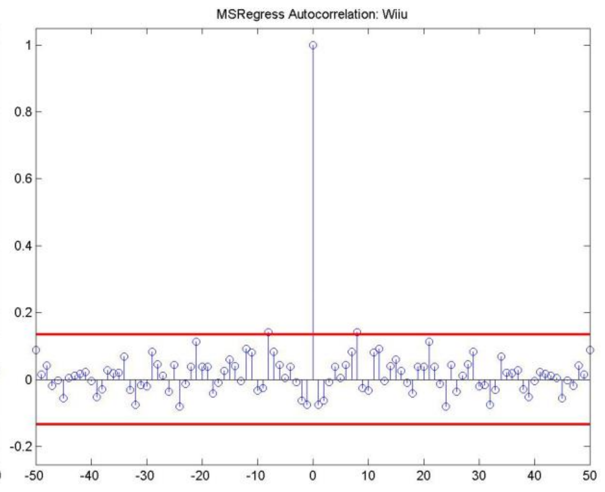

Fig. 27. MSRegress Wiiu Residuals

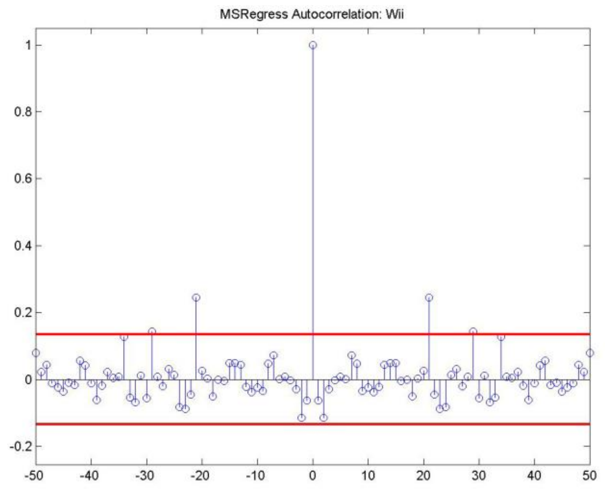

Fig. 28. MSRegress Wii Residuals

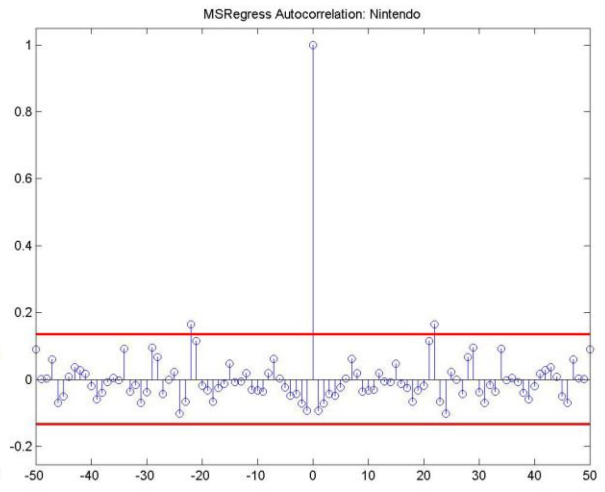

Fig. 29. MSRegress Nintendo Residuals

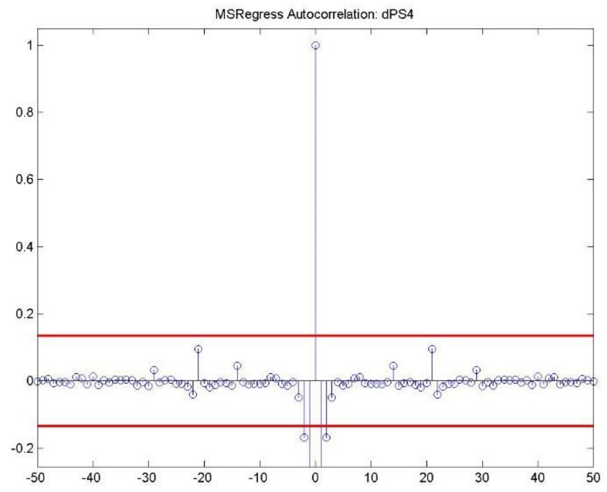

Fig. 30. MSRegress dPS4 Residuals

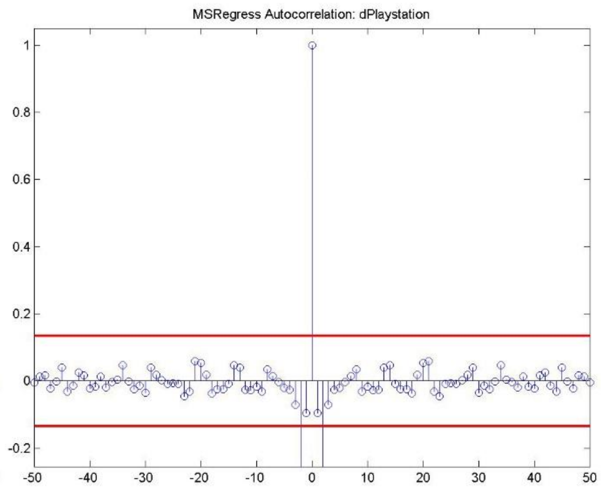

Fig. 31. MSRegress dPlaystation Residuals

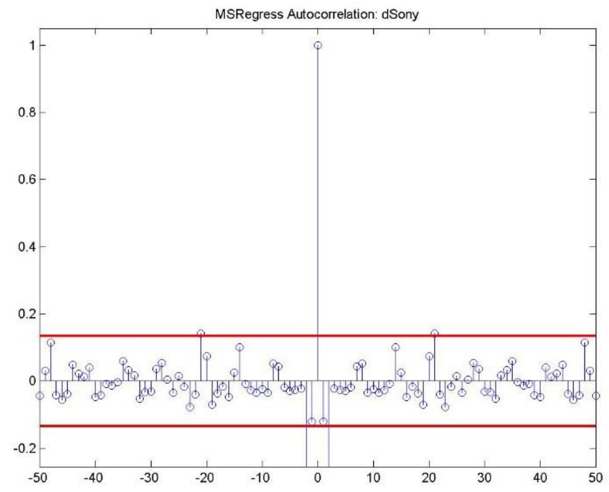

Fig. 32. MSRegress dSony Residuals

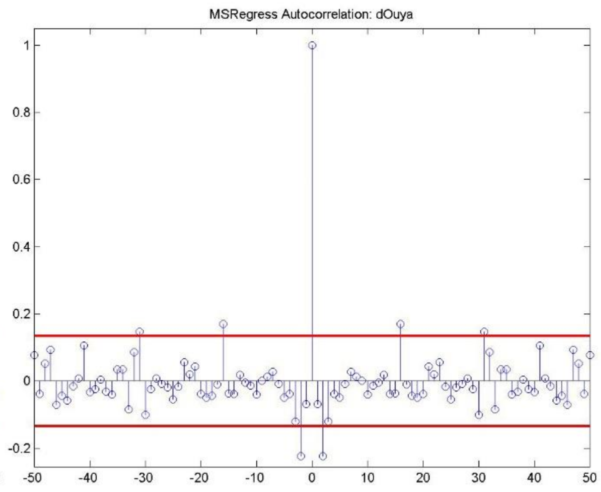

Fig. 33. MSRegress dOuya Residuals

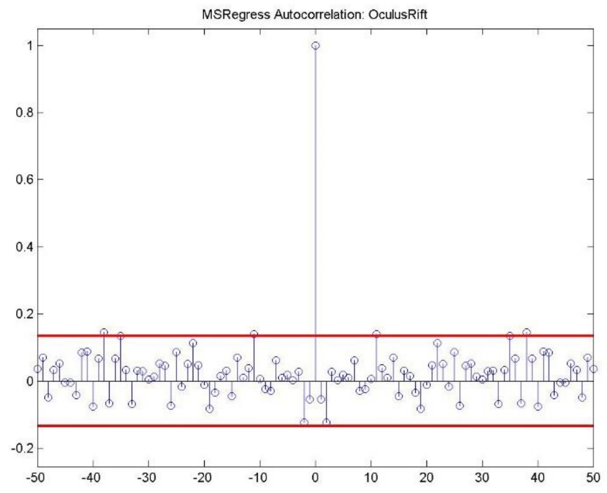

Fig. 34. MSRegress OculusRift Residuals

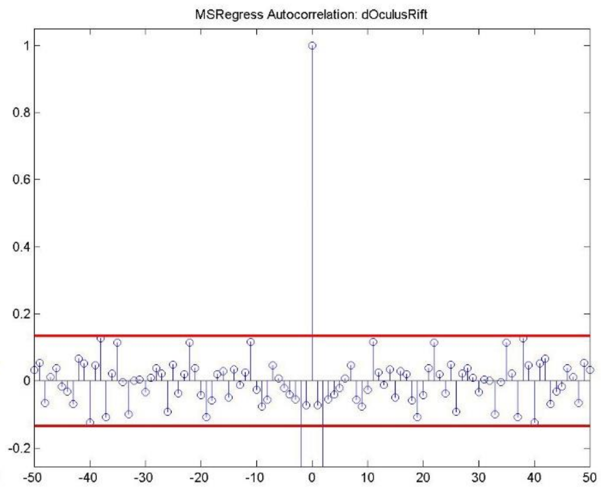

Fig. 35. MSRegress dOculusRift Residuals

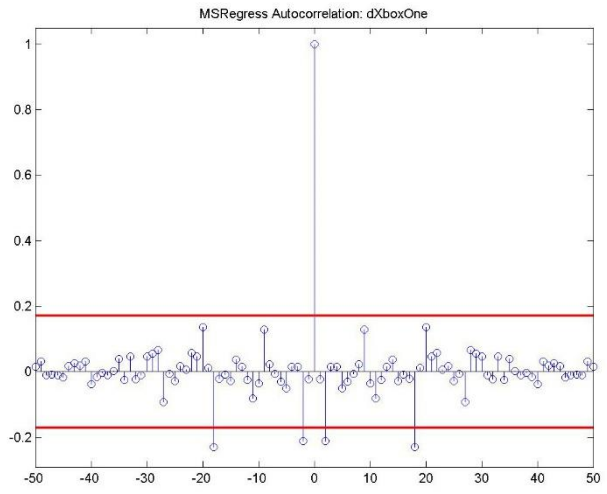

Fig. 36. MSRegress dXboxOne Residuals

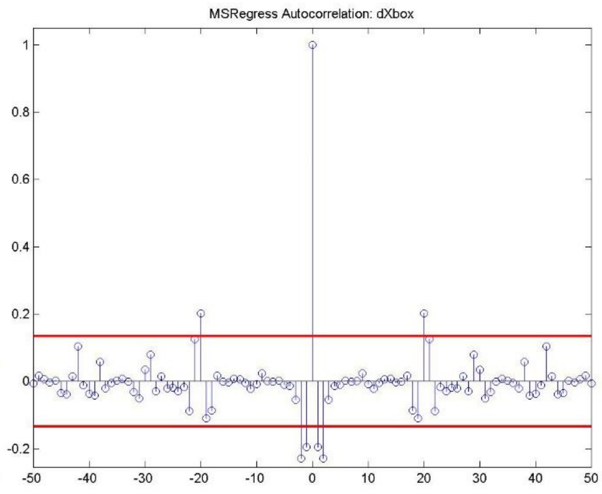

Fig. 37. MSRegress dXbox Residuals

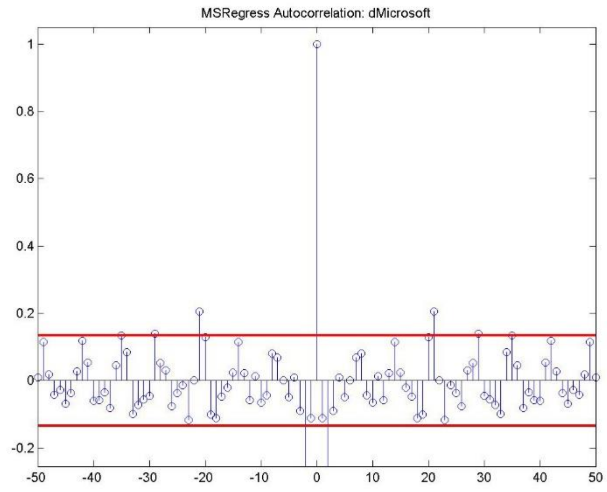

Fig. 38. MSRegress dMicrosoft Residuals

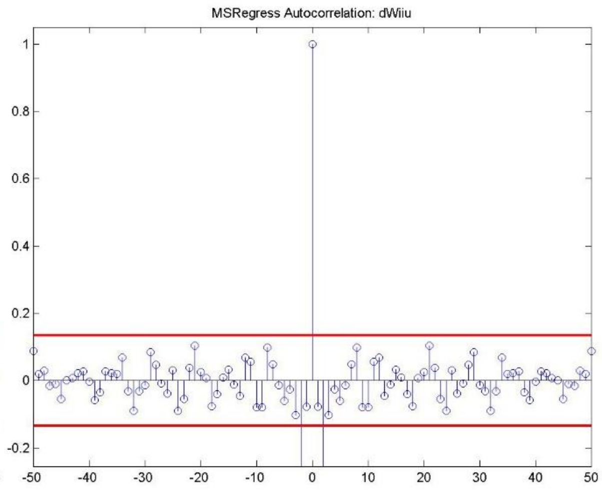

Fig. 39. MSRegress dWiiu Residuals

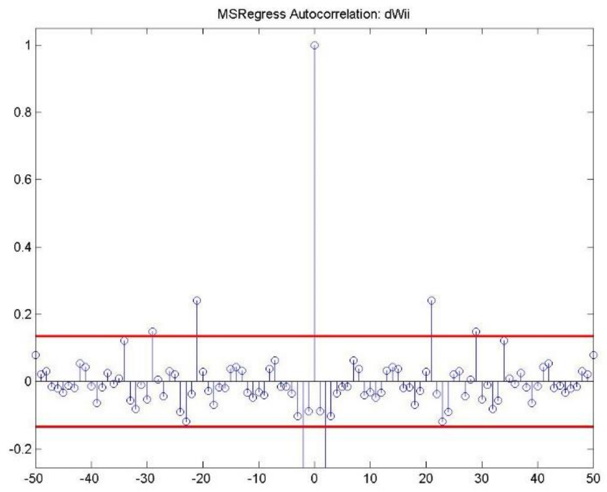

Fig. 40. MSRegress dWii Residuals

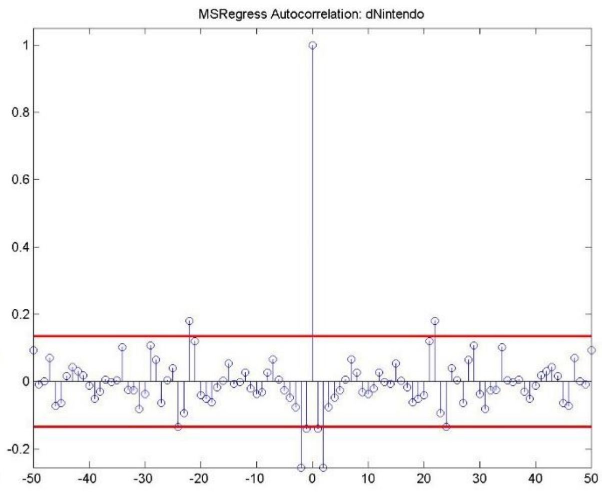

Fig. 41. MSRegress dNintendo Residuals

APPENDIX F: ARMA Residuals

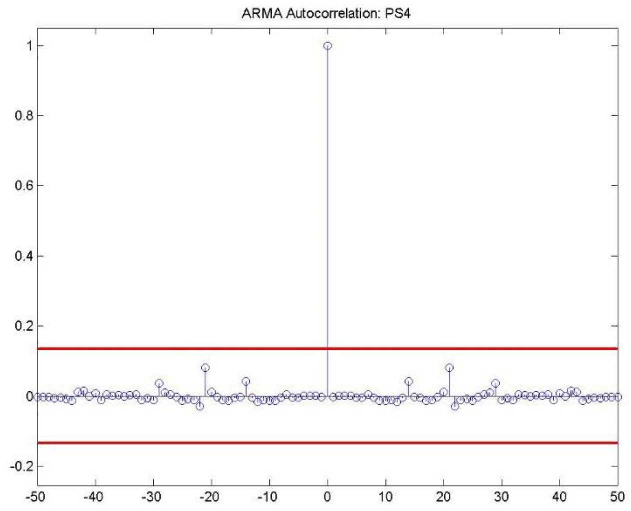

Fig. 42. ARMA PS4 Residuals

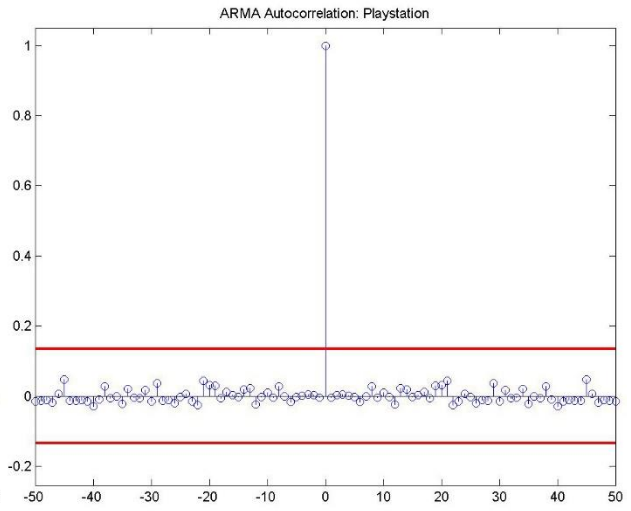

Fig. 43. ARMA Playstation Residuals

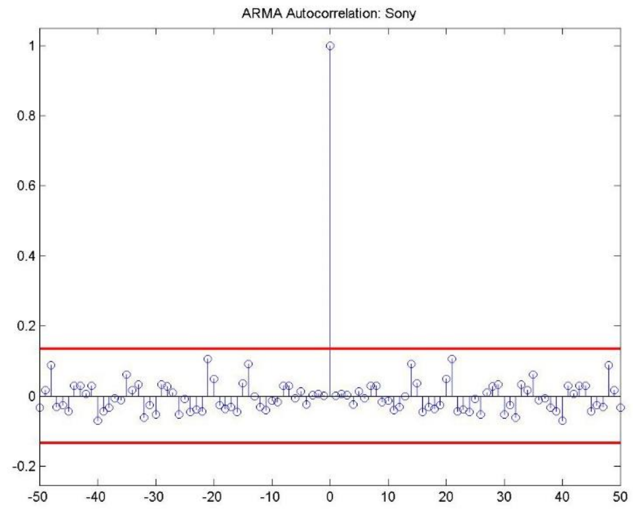

Fig. 44. ARMA Sony Residuals

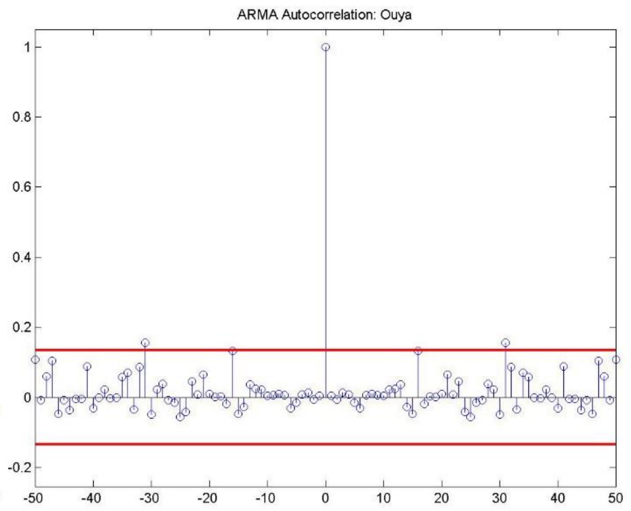

Fig. 45. ARMA Ouya Residuals

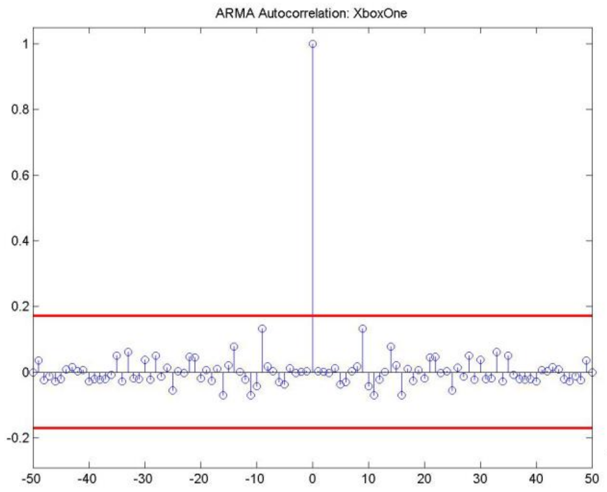

Fig. 46. ARMA XboxOne Residuals

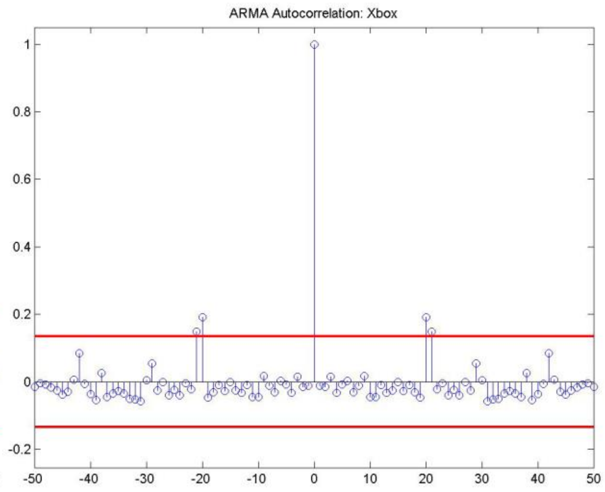

Fig. 47. ARMA Xbox Residuals

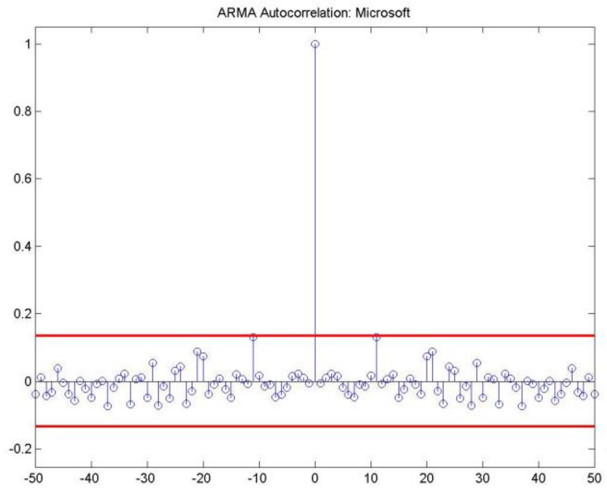

Fig. 48. ARMA Microsoft Residuals

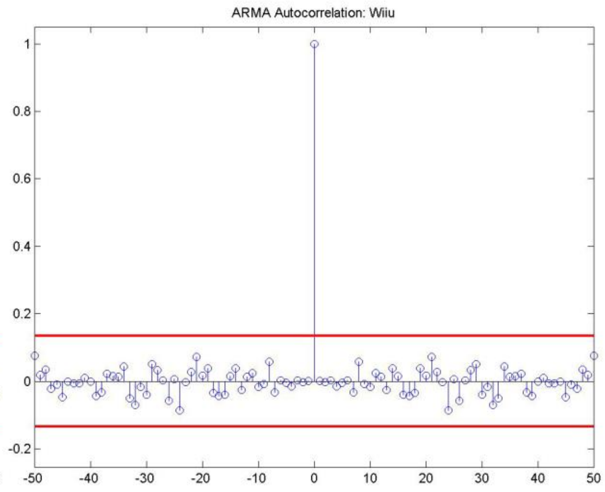

Fig. 49. ARMA Wiiu Residuals

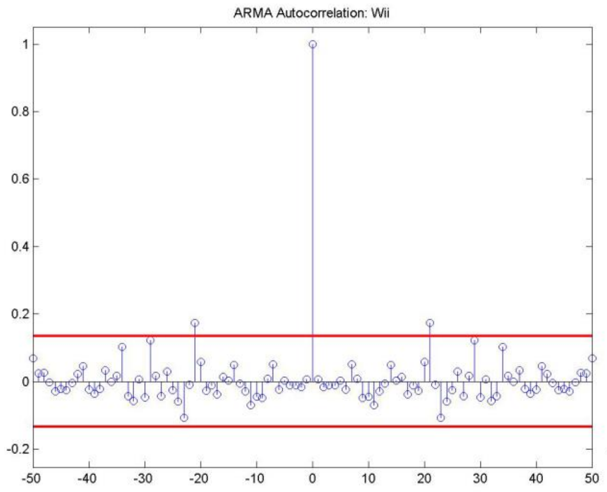

Fig. 50. ARMA Wii Residuals

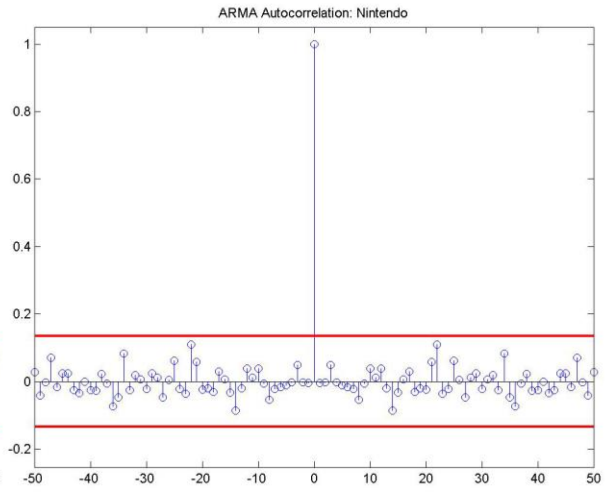

Fig. 51. ARMA Nintendo Residuals

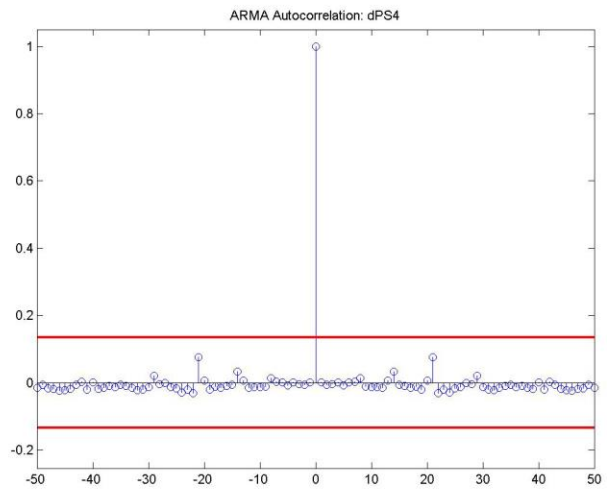

Fig. 52. ARMA dPS4 Residuals

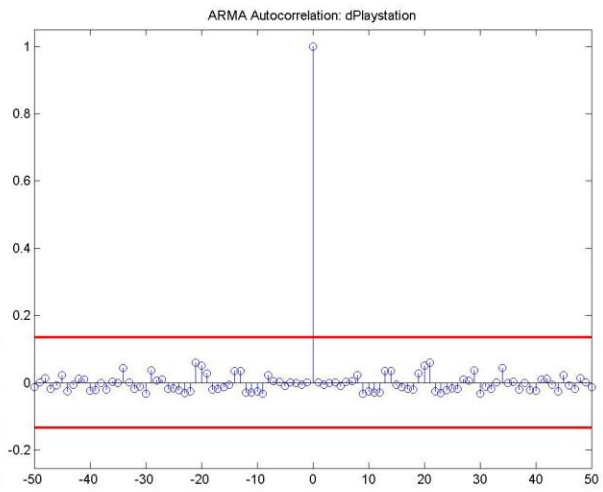

Fig. 53. ARMA dPlaystation Residuals

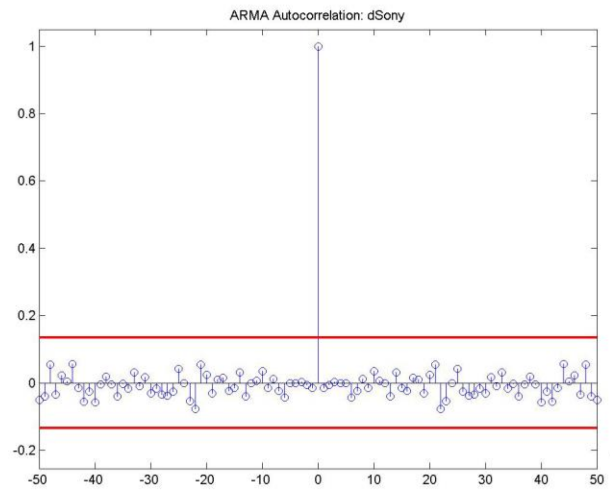

Fig. 54. ARMA dSony Residuals

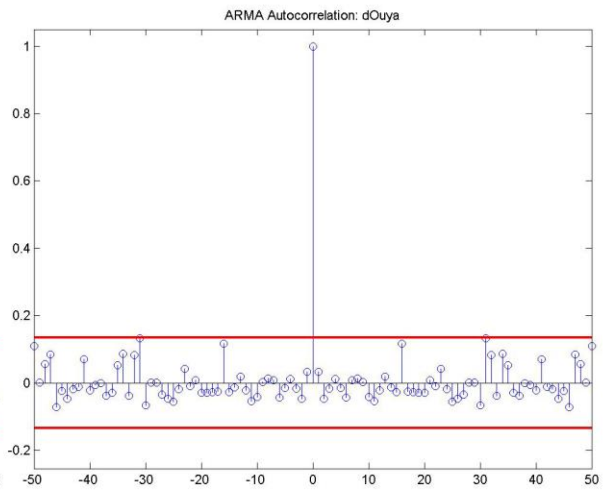

Fig. 55. ARMA dOuya Residuals

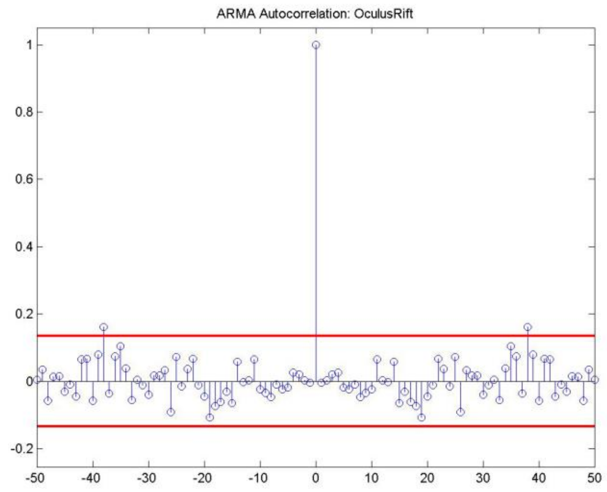

Fig. 56. ARMA OculusRift Residuals

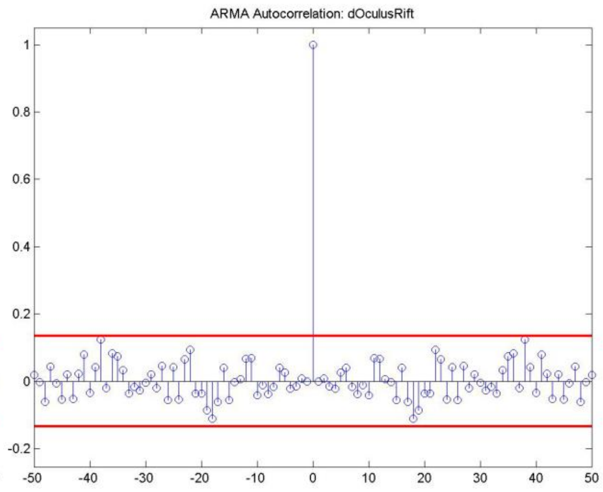

Fig. 57. ARMA dOculusRift Residuals

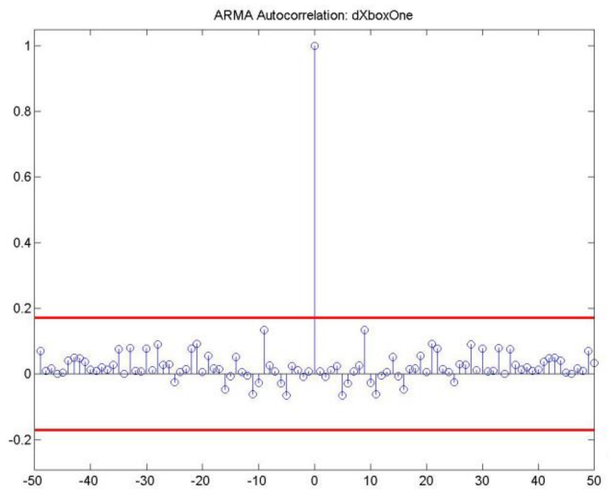

Fig. 58. ARMA dXboxOne Residuals

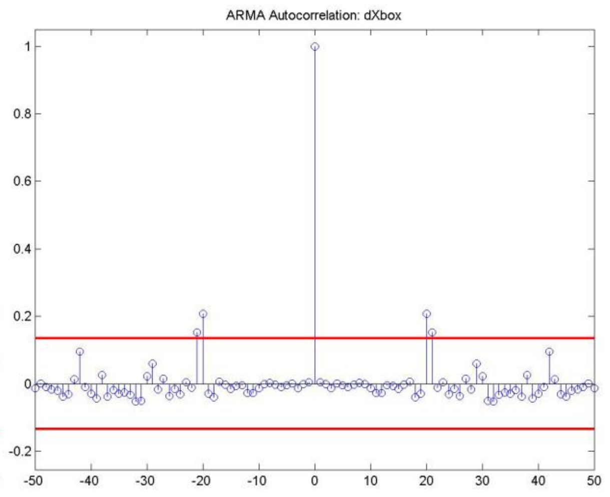

Fig. 59. ARMA dXbox Residuals

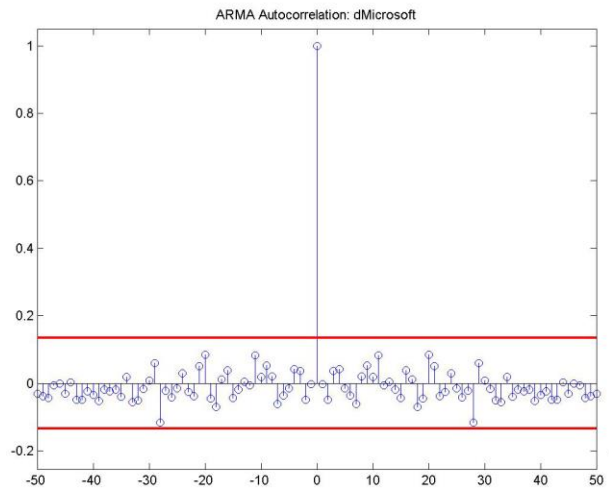

Fig. 60. ARMA dMicrosoft Residuals

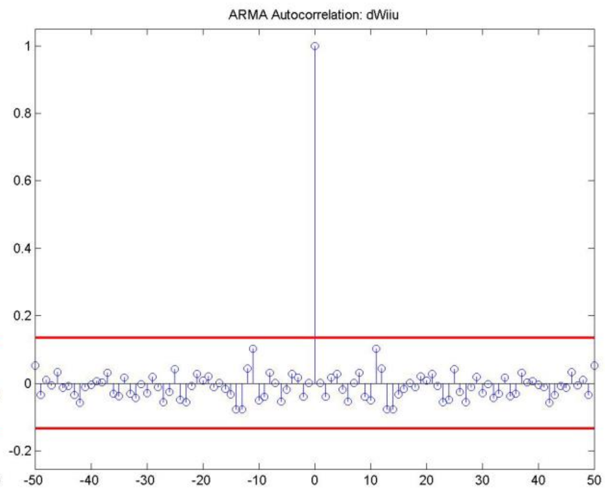

Fig. 61. ARMA dWiiu Residuals

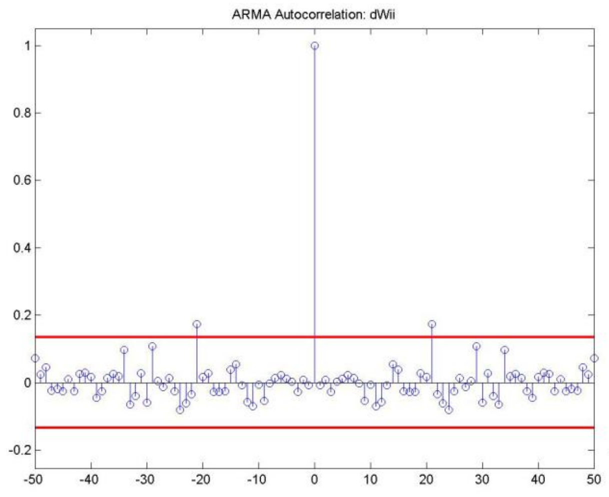

Fig. 62. ARMA dWii Residuals

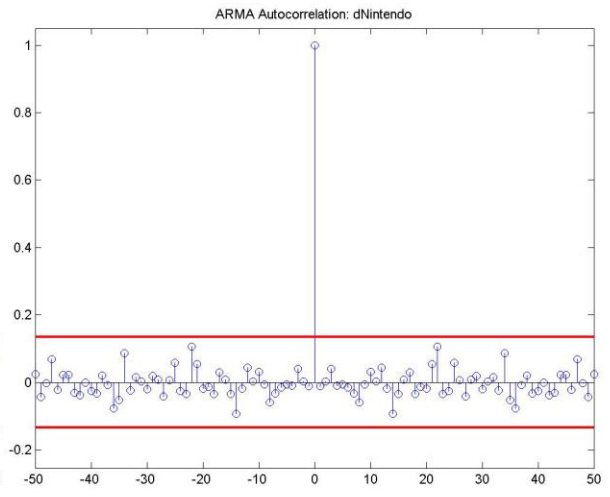

Fig. 63. ARMA dNintendo Residuals