Musicianship and Mandarin Chinese Tone Pronunciation

Jason Kao
June 1, 2017

In tone languages such as Mandarin Chinese and Vietnamese, word meaning is not only determined by lexical pronunciation, but it is also determined by the manipulation of pitch in every character. In Mandarin Chinese, there are four patterns (called tones) in which pitch can be manipulated: The first tone is high and level; the second starts low and moderately rises; the third falls and then rises again; the fourth starts high, but drops sharply to the bottom of the tonal range. This study aims at verifying if the production of lexical Mandarin tones varies as a function of musical aptitude and musical background. Thirty-one students taking their third year of Mandarin at Stuyvesant High School completed a survey for musical exposure and a musical battery for musical aptitude. They then submitted recordings of their voice which underwent a pitch analysis process to measure Mandarin tonal ability. Initial results show a moderate positive correlation between one's musical background and aptitude in tuning and one's ability to pronounce Mandarin tones, but more rigorous analysis is required.

I. INTRODUCTION

Pitch plays an important role in both language and music. In language, instead of conveying lexical information, it conveys information about speaker stance and emotion (T. Kumar, R. Kumar, & V. Kumar, 2013). However, in a class of languages known as tone languages, words assume different meanings depending on the tones (patterns of pitch fluctuation) in which they are spoken. In Mandarin Chinese, the syllable /ma/ can be associated with at least four different unrelated meanings based on the tone used (Creel, Weng, Fu, Heyman, & Lee, 2017). The four tones of Mandarin Chinese are manifested by distinctive pitch contours (Figure 1.1). The first tone is high and level; the second starts low and moderately rises; the third falls and then rises again; the fourth starts high, but drops sharply to the bottom of the tonal range (Chun, Jiang, & Avila, 2012).

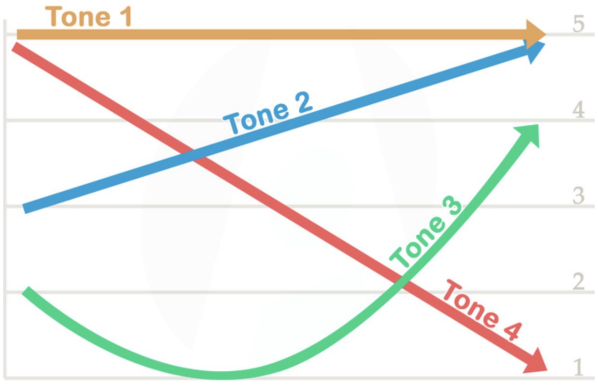

Figure 1.1 A textbook depiction of the four tones of Mandarin

Pitch is also a vital property of music, as it conveys information about melody and

tonality (the character of the piece as determined by which key it is played in) (Krumhansl, 1990).

Given the importance of music and tone in tonal languages, there emerges an intriguing question:

> Does the ability to pronounce tones in Mandarin Chinese increase with musical background and aptitude?

II. BACKGROUND

In 2010, Milovanov et al. attempted to find a correlation between the ability to discriminate between different pitch intervals in music and the ability to pronounce a foreign language. The subjects, all Finnish learners of English, were tested on their production of English phonemes and discrimination of phonemic minimal pairs. (Phonemic awareness is a phonological awareness in which listeners can identify between phonemes, the smallest units of sound that can differentiate meaning (Sensenbaugh, 1996)). Milovanov used phonemic minimal pairs for pitch discrimination and chord discrimination tests. He found no connection between the performance on phoneme discrimination tests and English production ability. However, performance on the English pronunciation test was better for subjects with a higher musical aptitude rather than those with a lesser one. This singular result reveals there may exist an observationally reproducible relationship between language pronunciation and musical aptitude.

Pfordresher & Brown (2009) further explored this relationship, but studied Eastern Asian languages (e.g. Mandarin, Chinese, Vietnamese), which are tone languages, rather than all foreign languages. He studied whether or not an individual's pitch ability was dependent on the use of pitch in the individual's native language. He used as a measure for overall pitch discrimination ability the summation of the ability to discriminate pitch intervals and the ability to imitate musical pitch through singing. Pfordresher & Brown discovered that tone language speakers, in comparison to non-tone language speakers, were significantly better at imitating musical pitch and discriminating between intervals of musical pitch. This result leads to my first hypothesis:

> H1: The greater the ability to discriminate between pitches, the greater the ability to pronounce tones in Mandarin.

Pitch is not the only aspect of musicality that is connected with foreign language pronunciation. People with greater abilities to create and detect melody differences also seem to perform better in L2 pronunciation. Delogu, Lampis, & Olivetti (2006) aimed to verify if the discrimination of lexical Mandarin tones varied with melodic ability. They presented subjects with no prior experience of any tone language with two lists of monosyllabic Mandarin words. The students were to identify whether the variation between two paired monosyllabic Mandarin words was phonological (in meaning) or tonal (in tone). Subjects who had a higher melodic ability were found to have performed significantly better in variation identification than their less musically talented counterparts. This leads to my second hypothesis:

> H2: The greater the melodic ability, the greater the ability to pronounce tones in Mandarin.

Not only does inherent musical aptitude play a part in foreign language pronunciation, but it also seems that musical training and experience positively affect foreign language pronunciation. Moreno et al. (2000) conducted a study to investigate brain plasticity and the effects of short periods of training. Over a six-month period, thirty-two non-musician children

were trained in several aspects of music (e.g. rhythm, melody, harmony). At the end of the training period, the children had significantly improved linguistic pitch processing. Moreno and his team's training these children for a relatively short period had strong consequences on the linguistic organization of the children's brain, demonstrating the fact that pitch processing in language was related to musical training. Supporting the findings Moreno et al. were Posedel, Emery, Souza, & Fountain. In their 2011 study, a greater number of years of musical experience was found to be significantly correlated with a higher pitch perception, and pitch perception was found to be a significant predictor of L2 pronunciation quality. These two studies lead to my third and fourth hypotheses:

H3: Tone pronunciation ability increases with training frequency.

H4: Tone pronunciation ability increases with musical experience.

In tandem with total hours of musical training per week, the type of instrument played should also be a factor of L2 pitch processing. Instruments which require fine tuning (violin, guitar, harp) require their players to pay more attention to closely the tone and pitch of their instrument. Most likely, subjects who play instruments that require fine tuning will perform better in L2 pronunciation than subjects who play instruments that do not require fine tuning. This leads to my fifth and sixth hypotheses:

H5: Playing a finely tuned instrument will increase tone pronunciation ability.

H6: The greater the tuning ability, the greater the ability to pronounce tones in Mandarin.

Regular exposure to music should also facilitate pitch processing, which leads to my last two hypotheses:

H7: L2 learners who listen to music on a regular basis will pronounce tones better.

H8: L2 learners who are involved in a professional group (band, orchestra, DJ) will pronounce tones better.

III. DATA COLLECTION

This study used a quantitative approach to investigate the musical predictors of tone pronunciation in Mandarin Chinese.

Participants

Thirty-one Stuyvesant High School students (eighteen male, thirteen female) participated in this study, all of whom are currently in their third year of Mandarin Chinese class.

Materials

Participants completed two surveys: a demographic survey to measure musical background (found in Appendix A), and a battery to measure musical aptitude. For the demographic survey, I consulted my statistics teacher to verify that there existed no leading questions. Moreover, commonly quantitative variables (i.e. how often one practices, how often one listens to music) were made to be categorical (displaying ranges of hours) in an attempt to eliminate selective memory and exaggeration in these self-reported data). Both the demographic and the musical aptitude surveys were shared with the students from the five Mandarin III sections of Stuyvesant

High School. Of the sixty-two survey responses, thirty-one were usable[1].

The musical test battery was the Profile of Musical Perception Skills (PROMS), which measures perceptual musical skills across multiple domains. (Based on prior literature, the participants were tested on their melodic, tuning, and pitch ability; these subtests are further explained below). Each of the domains' tests contained ten questions. The questions each consisted of two sections: a standard stimulus and a comparison stimulus. Participants were asked to choose on a Likert scale[2] how the standard and comparison stimuli compared. (Law & Zentner, 2012)

Melody. All melodic stimuli were monophonic and were composed of a constant rhythm of eight notes. The difficulty of the trials was determined by note density and level of atonality (atonal music is music that seems to lack a clearly defined center ("Atonal")). Figure 3.1 shows examples of easy and complex melody structures. *Tuning.* Each tuning stimulus consisted of C4, E4, G4, and C5, forming a chord of 1.5 seconds in length. The difficulty levels of the stimuli were varied by subtle manipulations to the E4 note. *Pitch.* The pitch subtest played several notes at a constant frequency and intensity. The difficulty level of this subtest was manipulated by varying the degree of pitch difference between the standard and comparison stimulus. (Law & Zentner, 2012)

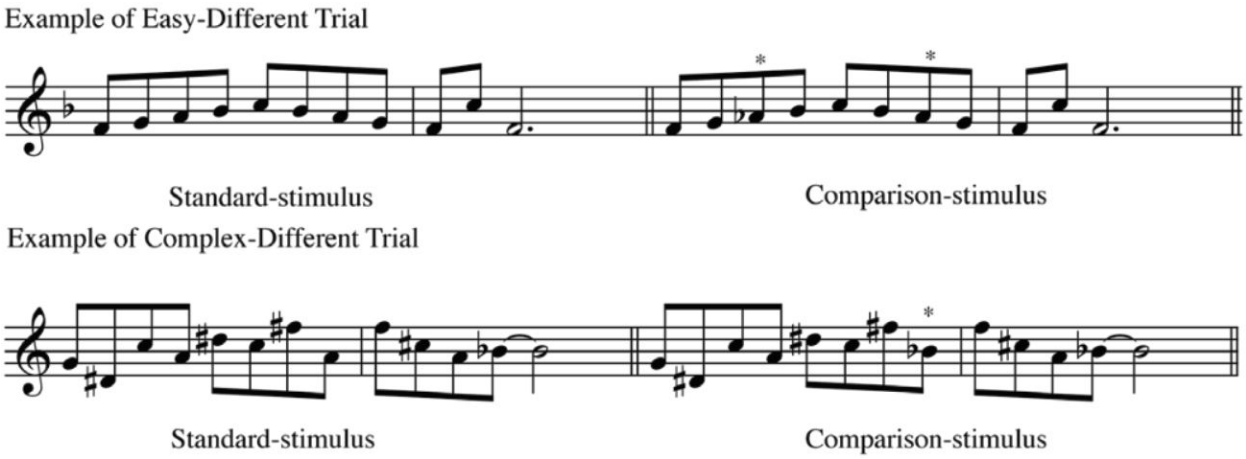

Figure 3.1 Above, an example of an easy comparison trial for melody; below, an example of a complex-different trial for melody. (Law & Zentner, 2012)

A participant's ability to pronounce tones in Mandarin Chinese was measured by comparing his or her audio to the audio of a native speaker. (Hui Zhu acted as the native speaker for this study; she is a Mandarin teacher who all participants have had for their three years of learning Mandarin). Every participant was recorded reading a list of four characters, one for each tone: 喝 (hē), 拿(ná), 好(hǎo), and 不(bù)[3]. These audio files are further analyzed in the next

[1] Most responses were valid. However, the website from which I hosted the survey saved all sessions. This means that opening the survey would automatically create a new user ID, despite the survey-taker not actually submitting the results. This created many survey responses that were almost or completely empty.

[2] A Likert scale was used in this study to minimize guessing and garner more accurate results (McLeod, 2008). The scale consisted of the choices "definitely different", "probably different", "I don't know", "probably same", and "definitely same" for comparing the standard stimulus and the comparison stimulus.

[3] Chun, Jiang, & Avila, 2013 analyzed the recordings of Mandarin students with these four characters. Moreover, all

subsection.

Procedure

For each participant's audio file, the pitch contour for each one of the four characters was extracted into an x-variable (time in seconds) and a y-variable (pitch in Hertz) with the acoustic analysis software Praat[4] (version 6.0.28; retrieved on March 23, 2017 from http://www.praat.org).

These frequency versus time plots all had two versions: that of the participant and that of the native speaker. (The former will hereinafter be known as the t_p-f_p plot, and the latter will hereinafter be known as the t_q-f_q plot). The values of the two plots were normalized. Time was scaled from zero to one, and frequency values were collectively subtracted by the median frequency of that plot. (It makes sense to normalize frequency (the lowness and highness of pitch) with the aforementioned method, as there was a mix of male and female participants). The t_p-f_p plot and the t_q-f_q plot were then approximated into one hundred points (the list of times was transformed into {0.01, 0.02, …, 0.99. 1.00}). For each approximated point T_i, where $0.01 \leq T_i \leq 1.00$, two frequencies (pitches) were taken: that of the t_p-f_p plot and that of the t_q-f_q plot. These two pitches were grouped into an (a_i, b_i) pair coordinate and graphed (see Figure 3.2). The R-squared[5] of this a-b plot's regression served as the measure of tone pronunciation ability (hereinafter referenced as the Partial Tone Production Index[6]).

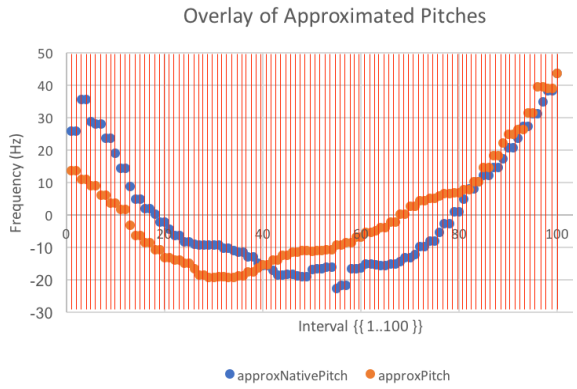

Figure 3.2(a) One hundred approximated values for the participant's frequency at T_i (a_i) and those for the native speaker's frequency at time T_i (b_i).

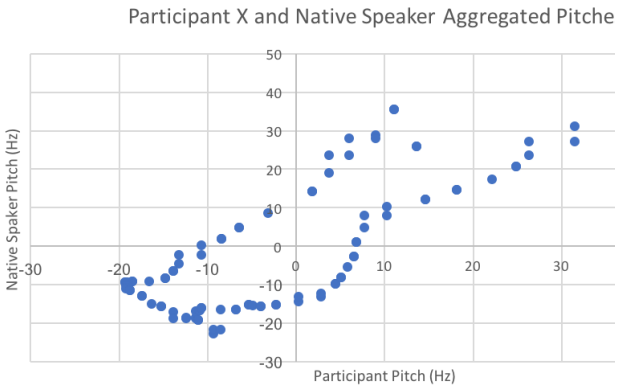

Figure 3.2(b) The scatter plot of the (a_i, b_i) pair coordinate.

The mean was taken from each participant's four Partial Tone Production Indices (one from each tone) to form a collective Tone Production Index, or TPI. Figure 3.3 shows the normal probability plot for the Tone Production Indices.

Mandarin students in their third year should be very comfortable with these characters' meanings and non-tonal pronunciation.

[4] Praat is a free, reliable software that has been used by previous literature, including Chun et al. (2013).

[5] Here it makes sense to use R-squared as the Partial Tone Production Index. If a student's pitches matched or were in a pattern with the native speaker's pitches, the R-squared would be high. If a student's pitches were not in the same pattern as the native speaker's pitches, the R-squared would be low.

[6] The code I wrote to complete this process can be found in Appendix F.

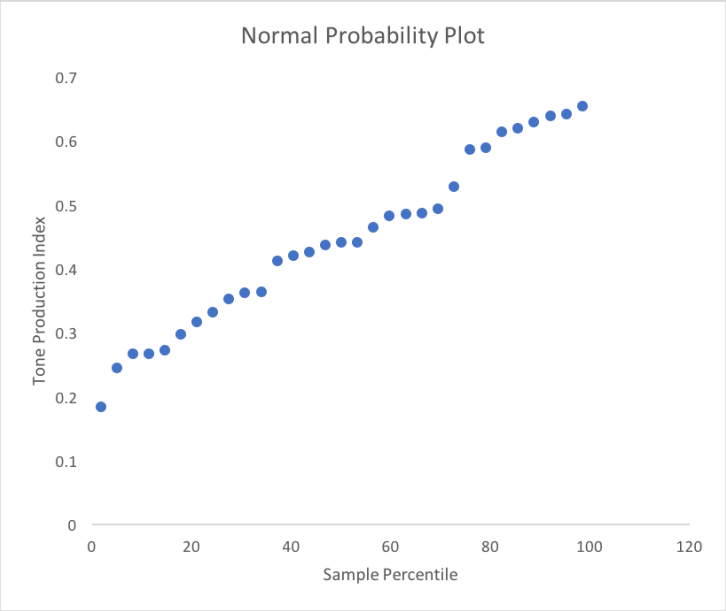

Figure 3.3 The normal probability plot for Tone Production Index is approximately linear, indicating the data are normally distributed.

IV. ANALYSIS

A multivariate regression was conducted to find if musical exposure and musical aptitude were significant predictors of the Tone Production Index (TPI). Excel was used to run this regression (version 15.33; retrieved on March 23, 2017 from https://products.office.com/en-us/excel).

There was no significant multicollinearity found (see Table 4.1). Normality and homoscedasticity of the stochastic error terms were checked.

	EXP	*FTI*	*PRC*	*DAY*	*GRP*	*STS*	*FAM*	*MEL*	*TUN*	*PIT*
EXP	1									
FTI	0.290	1								
PRC	0.383	0.335	1							
DAY	0.047	-0.271	-0.01	1						
GRP	0.338	-0.155	0.334	0.018	1					
STS	0.461	0.186	0.304	0.197	0.131	1				
FAM	0.184	-0.155	0.410	-0.019	0.385	0.444	1			
MEL	0.555	0.212	0.100	0.058	0.028	0.28	0.128	1		
TUN	0.422	-0.168	-0.055	0.184	0.106	0.037	-0.04	0.379	1	
PIT	0.151	-0.283	-0.062	0.257	-0.126	0.035	0.156	-0.096	0.488	1

Table 4.1: Correlation matrix
(unabbreviated output can be found in Appendix B)

	Expected Sign	*Coefficient (Std Error)*	*Adjusted p-value*[7]
intercept		0.34468 (0.13941)	0.011257929
musical experience	positive	-0.00041 (0.00775)	0.479118969
finely tuned instrument?	positive	0.1242 (0.07421)	0.054898947*
practice per week	positive	0.00995 (0.05859)	0.433416219
daily exposure to music	positive	0.00288 (0.02291)	0.450593433
involved in pro group?	positive	-0.12816 (0.13263)	0.172716349
musical status	positive	-0.01547 (0.04738)	0.373700048
family's musical status	positive	-0.00182 (0.07111)	0.489934356
PROMS melody subtest	positive	-0.00884 (0.01004)	0.194713267
PROMS tuning subtest	positive	0.01888 (0.00951)	0.030494889**
PROMS pitch subtest	positive	0.00316 (0.01471)	0.416161095

R-squared	0.2483150550	*, **, *** indicate significant levels at the 90%, 95%, and 99% respectively.
Adjusted R-squared	0.1946232732	
No. Observations	31	

Table 4.2: Regression Output
(full output can be found in Appendix C)

The model had an R-squared of 0.37998 and an Adjusted R-squared of 0.06998. The F-statistic [$F(10,20) = 1.22572$, $p = 0.33343$] revealed that the combination of the above variables did not have validity in fitting the data.

To better the fit of the model, the variables were down-selected. It was found that if a participant played a fine-tuned instrument (e.g. violin, guitar) was a predictor of TPI at the 90% significance level ($\beta_1 = 0.12$, $p < 0.10$), and tuning ability was a predictor of TPI at the 95% significance level ($\beta_1 = 0.02$, $p < 0.05$). Although the only significant variable of the PROMS test was tuning, a review of the literature had indicated that both PROMS melody and PROMS pitch should have also been significant variables. PROMS melody and PROMS pitch line fit plots are plotted in Figure 4.1a and 4.1b, respectively.

[7] The p-value's of an Excel regression output are representative a two-tailed result, so they were "adjusted" (divided by two).

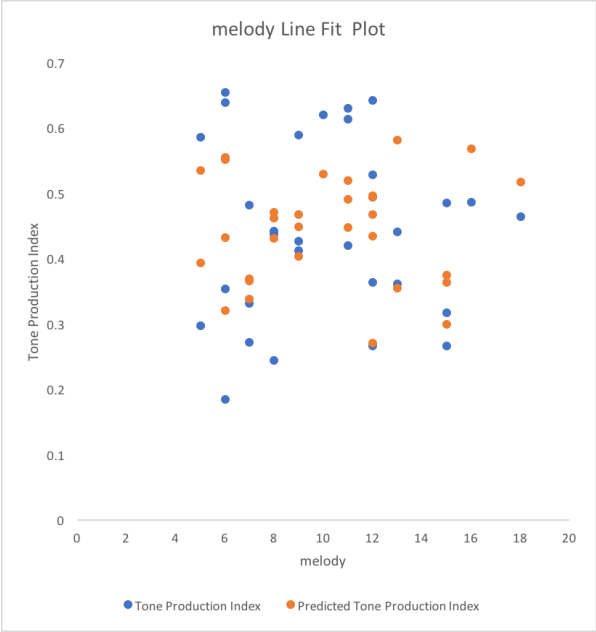

Figure 4.1a. It is clear from the line fit plot that there exists no correlation between melodic ability and Tone Production Index.

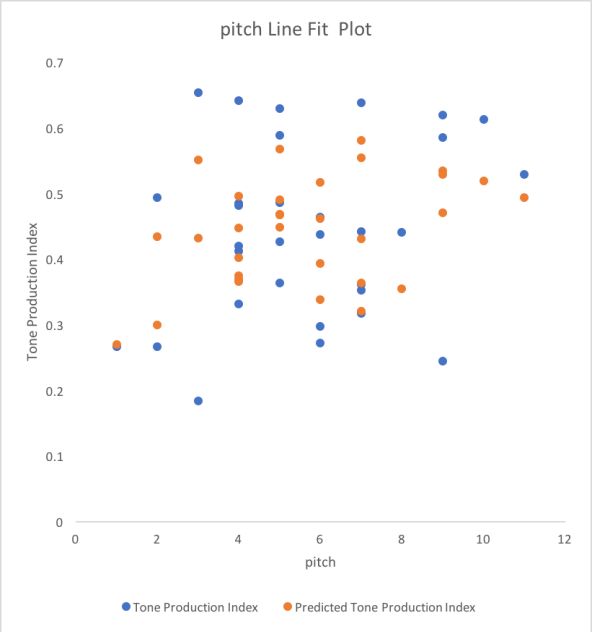

Figure 4.1b. It is clear from the line fit plot that there exists no correlation between pitch ability and Tone Production Index.

Because the line fit plots for PROMS melody and PROMS pitch showed no correlation with TPI, those variables were not chosen in the down selecting. Moreover, despite professional musical group involvement being nearly significant, only two participants admitted to being involved in a professional musical group, rendering the variable useless. In conclusion, the variables which were chosen for a better model were if a participant played an instrument which required fine tuning and the score a participant received on the tuning subtest of the PROMS. The new regression is shown below.

	Expected Sign	*Coefficient (Std Error)*	*Adjusted p-value*
intercept		0.29434 (0.05678)	1.67592E-05
fineTuning?	positive	0.10686 (0.05032)	0.021352362**
tuning	positive	0.01517 (0.00606)	0.009221139***

R-squared	0.37998361	*, **, *** indicate significant levels at the 90%, 95%, and 99% respectively.
Adjusted R-squared	0.069975415	
No. Observations	31	

Table 4.3: Downselected Regression Output
(full output can be found in Appendix D)

After down selecting the variables of the model, the fit of the model increased significantly [$F(2,28) = 4.62482$, $p = 0.01839$]. Therefore, at a 95% significance level, there is strong statistical evidence that this model has validity in fitting the data. There was no pattern in the residuals (Figure 5.2).

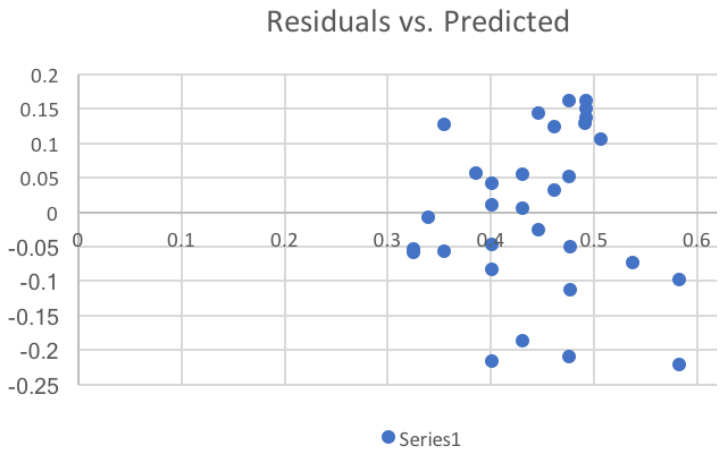

Figure 5.2. There exists no pattern within the residuals of the down-selected regression.

If a participant played a fine-tuned instrument became a more significant predictor (β_1 = 0.11, $p < 0.05$) as did tuning ($\beta_1 = 0.015$, $p < 0.01$). An equation for TPI can now be written:

Tone Production Index = 0.29434 + 0.10686(playsFinelyTunedInstrument?)
+0.015172(PROMSTuningAbility)

In summary, the multiple regression results indicate that the ability to pronounce tones in Mandarin Chinese is supported by playing a finely tuned instrument and increases with tuning ability.

V. DISCUSSION

This paper presented an analysis of tone pronunciation and musical aptitude with the goal of finding a correlation between the various properties of musical aptitude/background and the ability of a Mandarin student to pronounce tones in Mandarin. It was found that musical aptitude in tuning and whether a participant played a finely-tuned instrument were both significant predictors of Mandarin tone pronunciation skill, lending truth to H5 and H6.

Despite significant research conducted on the positive relationship between various properties of musical aptitude (melody, pitch) and tonal ability (Pfordresher & Brown, 2009; Milovanov et al., 2010; Delogu, Lampis, & Olivetti, 2006), this initial study did not suggest any such relationship, falsifying my first and second hypotheses, H1 and H2. And contrary to the studies of Moreno et al. (2000) and Posedel et al. (2011), musical training was not a significant predictor of L2 pronunciation quality, falsifying H3 and H4. Building from the two aforementioned studies was my belief that regular exposure to music should facilitate pitch processing and tone pronunciation. However, these variables were also deemed insignificant, falsifying H7 and H8.

There exist several limitations of this current study. The first is in the data collection process. The environment was not controlled for the recording of participant audio files and the completing of PROMS audio tests, in which the differences between standard and comparison stimuli are very sensitive. The fact that some users' environments may have been silent while others may have been very noisy contribute to the quality of the data collected. And the natural error in analyzing the pitch of audio files, especially in an uncontrolled environment, to some extent perverted the results of the regression. Further studies are advised to establish a standard recording room and microphone.

Moreover, there may exist voluntary bias in the survey. More students who are confident in their musical ability took the test than those who are not.

Furthermore, there were several questions on the survey which required self-reported data (how often a participant practices his or her instrument and how often a participant listens to music). Because self-reported data can rarely be independently verified, it may have contained several potential sources of bias. The most obvious of such sources is selective memory, in which participants incorrectly remember the amount of time they practice or listen to music. Due to the fact that many students listen to music while completing homework assignments, they may lose track of the extent of the time they listen to music.

Additionally, this study's small sample size of thirty-one was most likely the reason for the difficulty in finding significant relationships from the data. The small sample size was due to the PROMS battery's being twenty to thirty minutes long, which prompted many participants to leave the website in the middle of the survey. In hindsight, this problem could have been avoided by reducing the number of subsets of PROMS the participants were required to take. This solution would require more thorough research on the domains of musical aptitude that would positively affect Mandarin tone pronunciation. In summary, further studies are suggested to gather a much large sample size.

The findings of this study are applicable to the Mandarin classroom. If music is incorporated into the Mandarin curriculum (for example, through class singing, in which everybody must tune with everybody else), then students may be able to pronounce lexical Mandarin tones more naturally.

In conclusion, initial results show trends between the aforementioned variables. But despite a few significant explanatory variables, due to a small sample size and an uncontrolled recording/listening environment, a more rigorous analysis is necessary.

VII. Appendix

A. PROMS Demographic Survey Questions
B. Unabbreviated Correlation Matrix
C. Full Regression Output
D. Full Downselected Regression Output
E. Raw Data
F. Code for Tone Production Index

A. Survey Questions

1. Do you play a musical instrument? If so, please list all instruments and the time you have played them for.
2. How often do you practice playing your instrument? (categorical: do not play, 0-5 hours, 6-12 hours, 13-30 hours, more than 30 hours)
2. How often do you listen to music? (categorical: never, occasionally, 1-2 times/week, 3-4 times/week, 5-6 times/week, everyday)
3. Are you involved actively in professional listening activities? (e.g. Conducting, Sound Engineering, Piano Tuning, Performing, DJ-ing, Music Perception Research and others) (binary: yes/no)
4. Would you consider yourself a(n): (categorical: non-musician, music-loving non-musician, amateur musician, semi-professional musician, professional musician)
5. Are any members of your family musicians? (categorical: no, yes—amateurs, yes—professionals)

B. Unabbreviated Correlation Matrix

	musicalExperien	fineTuning?	practiceTime?	do you listen	Sound Enginee	musicality	amilyMusicalit	melody	tuning	pitch
musicalExperi	1									
fineTuning?	0.29015959	1								
practiceTime?	0.38336523	0.33524932	1							
How often do	0.04692331	-0.27086817	-0.00990148	1						
Are you involv	0.33836425	-0.15488062	0.33438812	0.01786854	1					
musicality	0.46126928	0.18626902	0.30351385	0.19721973	0.13145556	1				
familyMusical	0.18424789	-0.15510917	0.40978293	-0.0190879	0.38510245	0.44377105	1			
melody	0.55493055	0.21227495	0.10000875	0.05757158	0.02814252	0.28031878	0.12833212	1		
tuning	0.42192287	-0.16761222	-0.05546607	0.18418859	0.10612117	0.03725898	-0.0396374	0.37937474	1	
pitch	0.15141119	-0.28256818	-0.06219321	0.25749272	-0.12582246	0.03538752	0.15574885	-0.09583364	0.48808586	1

C. Full Regression Output

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.61642811
R Square	0.37998361
Adjusted R S	0.06997542
Standard Err	0.12989235
Observations	31

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	10	0.20680397	0.0206804	1.22572118	0.33343752
Residual	20	0.33744048	0.01687202		
Total	30	0.54424445			

	Coefficients	*Standard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.34468478	0.13940815	2.47248668	0.02251586	0.05388449	0.63548508	0.05388449	0.63548508
musicalExper	-0.0004109	0.0077495	-0.0530252	0.95823794	-0.0165761	0.01575426	-0.0165761	0.01575426
fineTuning?	0.12419844	0.07421465	1.67350293	0.10979789	-0.0306106	0.27900749	-0.0306106	0.27900749
practiceTime	0.00995161	0.05858972	0.16985255	0.86683244	-0.1122644	0.13216763	-0.1122644	0.13216763
How often d	0.00288048	0.02290679	0.12574803	0.90118687	-0.0449022	0.0506632	-0.0449022	0.0506632
Are you invo	-0.1281579	0.13262699	-0.9663034	0.3454327	-0.404813	0.14849714	-0.404813	0.14849714
musicality	-0.0154733	0.04738392	-0.326551	0.7474001	-0.1143144	0.08336786	-0.1143144	0.08336786
familyMusica	-0.0018169	0.0711094	-0.025551	0.97986871	-0.1501485	0.14651469	-0.1501485	0.14651469
melody	-0.0088353	0.01004258	-0.8797819	0.38942653	-0.0297837	0.01211318	-0.0297837	0.01211318
tuning	0.01887994	0.00950964	1.98534733	0.06098978	-0.0009568	0.0387167	-0.0009568	0.0387167
pitch	0.00315548	0.01471024	0.21450897	0.83232219	-0.0275295	0.03384049	-0.0275295	0.03384049

D. Downselected Regression Output

SUMMARY OUTPUT								
Regression Statistics								
Multiple R	0.4983122							
R Square	0.2483151							
Adjusted R S	0.1946233							
Standard Err	0.1208748							
Observations	31							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	2	0.1351441	0.067572	4.6248243	0.0183866			
Residual	28	0.4091004	0.0146107					
Total	30	0.5442444						
	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.2943445	0.0567765	5.1842712	1.676E-05	0.1780432	0.4106458	0.1780432	0.4106458
fineTuning?	0.106856	0.0503264	2.1232608	0.0427047	0.0037671	0.2099449	0.0037671	0.2099449
tuning	0.0151727	0.0060629	2.502558	0.0184423	0.0027535	0.027592	0.0027535	0.027592

E. Raw Data

Survey and PROMS:

id	EXP	FTI	PRC	DAY	GRP	STS	FAM	MEL	TUN	PIT
3	0	0	0	5	0	1	0	5	4	6
4	9	0	0	5	0	2	0	15	9	4
5	14	0	1	3	1	2	1	15	12	2
7	0	0	0	5	0	0	0	6	12	7
8	18	0	0	5	0	2	0	11	14	10
9	0	0	0	3	0	1	0	9	7	4
10	0	0	0	5	0	1	0	8	9	6
11	9	1	0	5	0	1	0	12	5	5
12	8	1	1	5	0	2	0	11	6	5
17	12	1	1	3	0	1	0	16	12	5
18	0	0	0	5	0	1	0	7	2	6
19	2.5	1	1	5	0	1	0	12	6	4
21	0	0	0	5	0	1	0	5	11	9
22	0	0	0	5	0	1	1	15	7	7
24	10	0	0	5	0	0	0	18	16	6
28	8	0	2	5	0	2	1	8	9	9
30	0	0	0	5	0	1	0	11	10	4
31	0	0	0	1	0	0	0	8	7	7
34	5	0	0	5	0	2	1	9	10	5
36	0	0	0	5	0	1	0	7	4	4
37	8	0	1	5	0	2	2	13	6	8
40	0	0	0	5	0	1	0	6	7	3
44	5	0	0	3	0	1	0	12	2	1

45	12.5	0	0	5	0	1	0	12	12	11
48	0	0	0	5	0	1	0	10	13	9
49	8	1	1	5	0	2	0	12	4	2
54	4	1	0	1	0	2	1	9	5	5
57	0	0	0	5	0	0	0	7	3	4
59	7	1	1	2	0	0	0	6	6	3
60	10	0	1	5	1	1	1	6	7	7
61	12	1	0	5	0	2	0	13	12	7

Unabbreviated variables from left to right: musical experience (in years), plays finely tuned instrument?, practice per week, music daily exposure, involved in professional group?, musical status, family musical status, PROMS melody score, PROMS tuning score, PROMS pitch score

Tone Analysis:

id	tone1	tone2	tone3	tone4	TPI (mean)
3	0.110181952	1.57E-04	0.756715196	0.324928393	0.297995674
4	0.007452586	0.858582327	0.170198212	0.905010304	0.485310857
5	0.169101207	0.856473347	0.035364266	0.007421853	0.267090168
7	0.049852734	0.959659635	0.705504978	0.839883848	0.638725299
8	0.244250879	0.853749523	0.372980667	0.982761739	0.613435702
9	0.388302788	0.718473798	0.114051164	0.427621676	0.412112356
10	0.309696429	0.496255278	0.015075831	0.928793001	0.437455135
11	0.082905377	0.087683581	0.428614372	0.857448507	0.364162959
12	0.075347477	0.834846137	0.680721542	0.927932187	0.629711836
17	0.005448186	0.91880273	0.072421785	0.948755344	0.486357011
18	7.66E-04	0.308778297	0.066241275	0.712589601	0.272093886
19	0.07626461	0.730622615	0.871278093	0.891004027	0.642292336
21	0.001623671	0.887267144	0.844113731	0.610880966	0.585971378
22	0.031850498	0.41826616	0.598274306	0.220592288	0.317245813
24	3.97E-04	0.89029924	0.002871472	0.964804688	0.464593201
28	0.037295888	0.864322174	0.054834951	0.023265306	0.24492958
30	0.002121153	0.787433358	0.010515779	0.880446575	0.420129216
31	0.036119684	0.842689576	0.014141526	0.874498141	0.441862232
34	0.046572406	0.857390262	0.532795472	0.920358022	0.58927904
36	0.003415016	0.862084213	0.216964316	0.846256491	0.482180009
37	0.088632488	0.74336642	0.072121336	0.86190222	0.441505616
40	0.040270338	0.300640896	0.002315727	0.39460021	0.184456793
44	0.030217794	0.840150372	0.066827475	0.131261296	0.267114234
45	0.239597026	0.769561914	0.136610931	0.970073127	0.528960749
48	0.023804435	0.899040757	0.676259329	0.881856082	0.620240151
49	0.018605938	0.921436037	0.350735371	0.685185794	0.493990785
54	0.027225867	0.651660514	0.075182132	0.952501245	0.42664244
57	0.013420404	0.560191852	0.003538517	0.751324793	0.332118891

59	0.532677921	0.828593205	0.351752508	0.903244699	0.654067083
60	0.015356147	0.418073082	0.792106524	0.188274994	0.353452687
61	0.163917649	0.409484361	0.004046177	0.870760611	0.362052199

F. Code for Tone Production Index (using Java 1.8.0_131)

```
import java.util.*;
import java.io.*;

public class Driver {

    private static double[][] toneStorer = new double[70][12];
    public static void main(String[] args)
    {
        File f0 = new File("/Users/jasonkao/Desktop/pitch-analyzer/f0");
        File[] files = f0.listFiles();
        File[] zhuFiles = {
            new
File("/Users/jasonkao/Desktop/pitch-analyzer/backup/zhu-tone1.txt"),
            new
File("/Users/jasonkao/Desktop/pitch-analyzer/backup/zhu-tone2.txt"),
            new
File("/Users/jasonkao/Desktop/pitch-analyzer/backup/zhu-tone3.txt"),
            new
File("/Users/jasonkao/Desktop/pitch-analyzer/backup/zhu-tone4.txt")
        };

        String dir = "/Users/jasonkao/Desktop/pitch-analyzer/f0/";
        for ( File f : files ) {
            if (f.getName().equals(".DS_STORE")) {
                continue;
            }
            try {
                String[] fileParts = f.getName().split( "-" );
                int userID = Integer.parseInt( fileParts[0] );
                int tone = fileParts[1].charAt(0) - '0';

                Scanner userScan = new Scanner( f );
                Scanner zhuScan = new Scanner( zhuFiles[ tone - 1 ]);

                User user = new User( userScan, zhuScan, userID == 348 && tone
== 3 );
                toneStorer[ userID ][ tone - 1 ] = user.getRSquared();
                toneStorer[ userID ][ 4 + tone - 1 ] = user.getRMSD();
                toneStorer[ userID ][ 8 + tone - 1 ] = user.getMAE();
            } catch (Exception e) {
                e.printStackTrace();
                System.out.println("!#EXCEPTION: Failed to read file " + f);
            }
        }
        printOut( toneStorer, 2 );
    }
    /*
     * @parameter n
     * 0-Rsquared
     * 1-RMSD
     * 2-MAE
     */
```

```
    public static void printOut(double[][] d, int n)
    {
System.out.println("userID,tone1,tone2,tone3,tone4");//,RMSD1,RMSD2,RMSD3,RMSD
4,MAE1,MAE2,MAE3,MAE4");
        for (int u = 0; u < d.length; u++) {
            if ( takeSum(d[u]) == 0.0 ) { // empty means 0.0
                continue;
            }
            System.out.print(u + ",");
            for (int i = n * 4; i < n * 4 + 4; i++) {
                System.out.print(d[u][i] + ",");
            }
            System.out.println();
        }
    }

    private static double takeSum(double[] d) {
        double output = 0.0;
        for (double dub : d) {
            output += dub;
        }
        return output;
    }

    public static void print2D(double[][] d)
    {
        String output = "";
        for (int i = 1; i < d.length; i++) {
            output += "[";
            for (double x : d[i]) {
                output += x + ", ";
            }
            output = output.substring(0, output.length() - 2) + "]\n";
        }
        System.out.println(output);
    }

    public static void print1D(Object[] o) {
        String output = "";
        for (Object x : o) {
            output += x + ", ";
        }
        if (o.length > 0) {
            output = output.substring(0, output.length() - 2);
        }
        System.out.println(output + "]");
    }
}

public class MyHeap {

    private int O_CONST;

    private Double[] ary;
```

```
    private int size;

    public MyHeap() {
        ary = new Double[10];
        size = 1;
        O_CONST = 1;
    }

    public MyHeap(boolean isMax) {
        this();
        if (isMax) {
            O_CONST = 1;
        } else {
            O_CONST = -1;
        }
    }

    public void add(Double i)
    {
        if (size == ary.length) {
            resize();
        }
        ary[size] = i;
        pushUp();
        size++;
    }

    private void pushUp()
    {
        int tracker = size;
        while (tracker > 1 && ary[tracker].compareTo( ary[tracker / 2] ) *
O_CONST > 0) {
            swap(tracker, tracker / 2);
            tracker = tracker / 2;
        }
    }

    private void resize()
    {
        Double[] biggerAry = new Double[ary.length * 2];
        for (int i = 0; i < ary.length; i++) {
            biggerAry[i] = ary[i];
        }
        ary = biggerAry;
    }

    public Double remove()
    {
        Double output = ary[1];
        ary[1] = ary[size - 1];
        pushDown();
        size--;
        return output;
    }
```

```
    private void pushDown()
    {
        int tracker = 1;
        while (2 * tracker + 1 < size) {
            boolean goLeft = ary[2 * tracker].compareTo(ary[2 * tracker + 1])
* O_CONST > 0;
            if (goLeft) {
                if (ary[tracker].compareTo( ary[2 * tracker] ) * O_CONST < 0)
{
                    swap(tracker, 2 * tracker);
                    tracker = 2 * tracker;
                } else if (ary[tracker].compareTo( ary[2 * tracker + 1] ) *
O_CONST < 0) {
                    swap(tracker, 2 * tracker + 1);
                    tracker = 2 * tracker + 1;
                } else {
                    break;
                }
            } else {
                if (ary[tracker].compareTo( ary[2 * tracker + 1] ) * O_CONST <
0) {
                    swap(tracker, 2 * tracker + 1);
                    tracker = 2 * tracker + 1;
                } else if (ary[tracker].compareTo( ary[2 * tracker] ) *
O_CONST < 0) {
                    swap(tracker, 2 * tracker);
                    tracker = 2 * tracker;
                }  else {
                    break;
                }
            }
        }
    }

    private void swap(int a, int b)
    {
        Double temp = ary[a];
        ary[a] = ary[b];
        ary[b] = temp;
    }

    public Double peek()
    {
        return ary[1];
    }

    public int size()
    {
        return size - 1;
    }

    public String toString()
    {
```

```
        String output = "[";
        for (int i = 0; i < size; i++) {
            output += ary[i] + ", ";
        }
        if (size > 1) {
            output = output.substring(0, output.length() - 2);
        }
        return output + "] " + O_CONST;
    }

}

public class Regression {

    private double[] x, y;
    private double r, a, b;

    private double ROUND_OFF = Math.pow( 10, 5 ); // round to {{ ROUND_OFF }}
digits

    public Regression(ArrayList<Double> times, ArrayList<Double> pitches)
    {
        if (times.size() != pitches.size()) {
            throw new IllegalArgumentException("ArrayLists must be of same
length.");
        }

        x = new double[times.size()];
        y = new double[pitches.size()];

        for (int i = 0; i < times.size(); i++) {
            x[i] = times.get(i);
            y[i] = pitches.get(i);
        }

        r = getCovariance() / (getStDevX() * getStDevY());
    }

    public double getRSquared()
    {
        return r * r;
    }

    private double getCovariance()
    {
        double xbar = 0.0;
        for (double d : x) {
            xbar += d;
        }
        xbar /= x.length;

        double ybar = 0.0;
        for (double d : y) {
            ybar += d;
```

```
    }
    ybar /= y.length;

    double covariance = 0.0;
    for (int i = 0; i < x.length; i++) {
        covariance += (x[i] - xbar) * (y[i] - ybar);
    }
    covariance /= x.length - 1;
    return covariance;
}

private double getStDevY()
{
    double xbar = 0.0;
    for (double d : y) {
        xbar += d;
    }
    xbar /= y.length;

    double stdev = 0.0;
    for (double d : y) {
        stdev += Math.pow(d - xbar, 2);
    }
    stdev /= y.length - 1;
    return Math.pow(stdev, 0.5);
}

private double getStDevX()
{
    double xbar = 0.0;
    for (double d : x) {
        xbar += d;
    }
    xbar /= x.length;

    double stdev = 0.0;
    for (double d : x) {
        stdev += Math.pow(d - xbar, 2);
    }
    stdev /= x.length - 1;
    return Math.pow(stdev, 0.5);
}

private double getSlope()
{
    return r * getStDevY() / getStDevX();
}

private double getIntercept()
{
    double xbar = 0.0;
    for (double d : x) {
        xbar += d;
    }
```

```
        xbar /= x.length;

        double ybar = 0.0;
        for (double d : y) {
            ybar += d;
        }
        ybar /= y.length;

        return ybar - xbar * getSlope();
    }

    private void print1D(double[] d) {
        String output = "[";
        for (double x : d) {
            output += x + ", ";
        }
        if (d.length > 0) {
            output = output.substring( 0, output.length() - 2 );
        }
        System.out.println( output + "]" );
    }

    public String toString()
    {
        return "b: " +
            Math.round(b * ROUND_OFF) / ROUND_OFF +
            ", R-squared: " +
            Math.round(getRSquared() * ROUND_OFF) / ROUND_OFF;
    }

}
public class RunningMedian {

    public MyHeap left, right;

    public RunningMedian() {
        left = new MyHeap(true);
        right = new MyHeap(false);
    }

    public void add(double v)
    {
        if (left.size() + right.size() == 0) {
            left.add(v);
        } else if (v * 1.0 > getMedian()) {
            right.add(v);
        } else {
            left.add(v);
        }

        if (right.size() >= left.size() + 2) {
            left.add(right.remove());
        } else if (left.size() >= right.size() + 2) {
            right.add(left.remove());
```

```
        }
    }

    public double getMedian()
    {
        if (left.size() == right.size()) {
            return (left.peek() + right.peek()) / 2.0;
        } else if (left.size() > right.size()) {
            return left.peek();
        }
        return right.peek();
    }

    public String toString()
    {
        return left + "\n" + right;
    }

}

public class User implements Comparable<User> {

    private static int SIZE = 100; // how much to approximate into

    private int id, tone;
    private ArrayList<Double> times, pitches;
    private ArrayList<Double> zhuTimes, zhuPitches;
    private ArrayList<Double> approxTime, approxPitch;
    private ArrayList<Double> approxZhuTime, approxZhuPitch;
    private Regression reg;
    private MeanError me;

    public User(Scanner in, Scanner zhu, boolean debug)
    {
        id = in.nextInt();
        tone = in.nextInt();

        approxTime = new ArrayList<Double>();
        approxPitch = new ArrayList<Double>();
        approxZhuTime = new ArrayList<Double>();
        approxZhuPitch = new ArrayList<Double>();

        times = new ArrayList<>();
        pitches = new ArrayList<>();
        move( times, pitches, in );
        normalizeGood( times );
        normalize( pitches );
        approximate( times, approxTime );
        approximate( pitches, approxPitch );

        zhuTimes = new ArrayList<>();
        zhuPitches = new ArrayList<>();
        move( zhuTimes, zhuPitches, zhu );
        normalizeGood( zhuTimes );
```

```
        normalize( zhuPitches );
        approximate( zhuTimes, approxZhuTime );
        approximate( zhuPitches, approxZhuPitch );

        if (debug) {
            /*
            for (int i = 0; i < SIZE; i++) {
                System.out.println(approxPitch.get(i) + "," +
approxZhuPitch.get(i));
            }
            System.out.println("48-3Times,48-3Pitches");
            for (int i = 0; i < times.size(); i++) {
                System.out.println(times.get(i) + "," + pitches.get(i));
                }*/
            System.out.println("48-3ApproxTimes,48-3ApproxPitches");
            for (int i = 0; i < approxTime.size(); i++) {
                System.out.println(approxTime.get(i) + "," +
approxPitch.get(i));
            }
            System.out.println("zhuAATimes,zhuAAPitches");
            for (int i = 0; i < approxZhuTime.size(); i++) {
                System.out.println(approxZhuTime.get(i) + "," +
approxZhuPitch.get(i));
            }/*
            System.out.println("approxPitch,approxZhuPitch");
            for (int i = 0; i < approxTime.size(); i++) {
                System.out.println(approxPitch.get(i) + "," +
approxZhuPitch.get(i));
                }*/
        }

        reg = new Regression( approxPitch, approxZhuPitch );
        me = new MeanError( approxPitch, approxZhuPitch );
    }

    public double getRMSD() {
        return me.getRMSD();
    }

    public double getMAE() {
        return me.getMAE();
    }

    private static void approximate( ArrayList<Double> x, ArrayList<Double> y
)
    {
        for (int i = 0; i < SIZE; i++) {
            int j = (int) ( ( (double) x.size() ) / SIZE * i);
            y.add( x.get( j ));
        }
    }

    private static void concatenate(ArrayList<Double> x, ArrayList<Double> y)
{
```

```
        for (Double d : y) {
            x.add(d);
        }
    }

    private static void move(ArrayList<Double> times, ArrayList<Double>
pitches, Scanner in )
    {
        boolean isPitch = false;
        while (in.hasNext()) {
            String v = in.next();
            if (isDouble(v)) {
                if (isPitch) {
                    pitches.add(Double.parseDouble(v));
                } else {
                    times.add(Double.parseDouble(v));
                }
                isPitch = !isPitch;
            }
        }
    }

    private static boolean isDouble(String value) {
        try {
            Integer.parseInt(value);
            return false;
        } catch (NumberFormatException e) {}

        try {
            Double.parseDouble(value);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    private static void normalizeGood(ArrayList<Double> x)
    {
        double min = Collections.min(x);
        double max = Collections.max(x);

        for (int i = 0; i < x.size(); i++) {
            x.set(i, (x.get(i) - min) / (max - min) );
        }
    }

    private static void normalize(ArrayList<Double> x)
    {
        Double mean = 0.0;
        for (Double d : x) {
            mean += d;
        }
        mean /= x.size();
        for (int i = 0; i < x.size(); i++) {
```

```
            x.set(i, x.get(i) - mean);
        }
    }

    public double getRSquared() {
        return reg.getRSquared();
    }

    public String toString() {
        return id + "-" + tone + "; " + reg;
    }

    public int compareTo(User o)
    {
        return (10 * id + tone) - (10 * o.id + o.tone);
    }
}
```