Using Tarsos and `music21` to Analyze Presidential Speech Patterns

MIT student

Abstract

I believed that over the course of their first term in office, presidents would increase the size of their pitch range, and that their range would shift towards lower frequencies. I found no strong correlation suggesting that their range increased in size, but there was there a trend suggesting that their range decreased in frequency.

Introduction

Tarsos is a program for analyzing Music Information Retrieval (MIR) features of music in a culture-independent way[3]. Specifically, it can analyze an mp3 file, extract a pitch histogram, and calculate things such as the best fit “scale” that is being used (using peak finding algorithms).

I was interested in how this software might be applied in less conventional ways. I recently took a class on public speaking, and one of the things we examined was the actual pitches that we were using when we spoke. Thinking of these together, I became curious about the speech patterns of various presidents during their first term in office.

While I have obviously never been the President of the United States, I feel that perhaps I can relate to what that first term must feel like. In my four years of college, I entered feeling ready and that people expected a lot from me. I soon realized that I wasn't ready at all, but that's okay because I had some time to get there. I matured a lot, and grew as a speaker, a leader, and as a student. I became more experienced and more confident at handling the various situations that arise in college. Similarly, I'm sure that the President feels ready to take office by the time he is elected. During the first years, he begins to realize exactly what the role entails. Towards the end, he is more confident in his position and in his stances, and he is ready to handle any of the various crises that arise during a four year term as president.

I thought that because the president would gain much experience (and confidence) speaking over the course of his presidency, that he would become monotonically less monotonous in his speech patterns. Specifically, I believed that the range of pitches that he employs in his speeches would increase in size as he becomes more confident and perhaps dramatic. Additionally, I hypothesized that as his initial excitement faded into confident experience, he would use more lower pitches and his pitch range would shift downwards.

Methodology

The first task was finding recordings to use of various presidential speeches. Fortunately, I found a large database online that contains recordings of presidential speeches going back to President Roosevelt[2]. Some presidents had only a few recordings relative to others though, so I had to decide how to choose speeches that would be comparable from president to president, and also spread fairly evenly throughout their first term. I decided to use the State of the Union Addresses because every president back to Kennedy has a recorded speech given every year after being sworn in for their first term. They all occurred in the same time of year, in the same location, to the same audience. They were also all roughly the same duration. The recording technologies for each president was roughly the same throughout his term, and I only compared each recorded State of the Union Address with those of the same president. I also found Tarsos to be relatively robust to noise caused by older recording techniques.

After downloading the relevant files, I ran them through the Tarsos analysis program and exported the pitch and pitch class histograms as CSV files. I experimented with different settings on the first set of recordings in order to find one set of options[1] that gave a near-optimal scale description for each of those three recordings. Once I found these options, I used the same set of options for each file and also extracted a Scala[4] file.

Finally, I used `music21`[5] as well as some of my own code (see Appendix B) to analyze the data and extract information about range, most common pitches, average interval between Scala pitch classes, etc. Of the various metrics that I analyzed, I decided that the most relevant were the highest and lowest used pitch, the average pitch, and the range (highest minus lowest pitch).

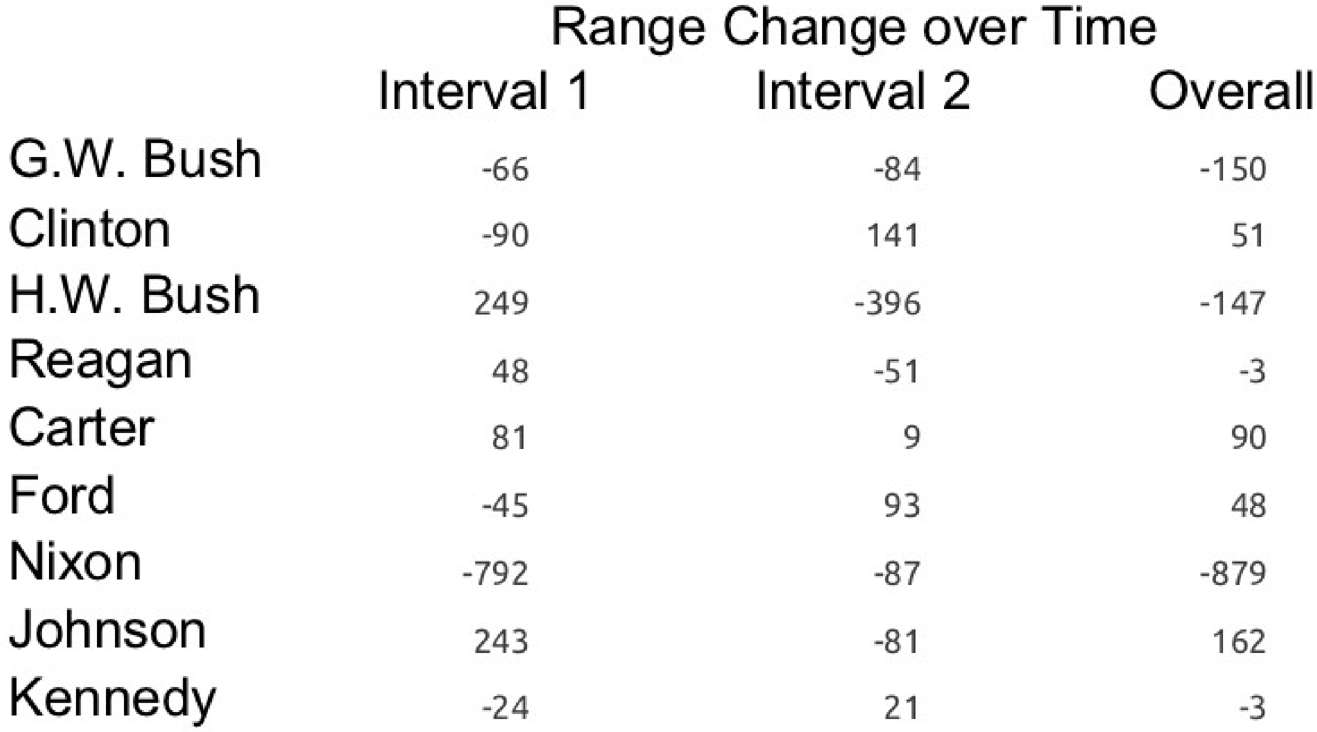
For the highest, lowest, and average pitch, I filtered the pitch histogram produced by Tarsos by number of annotations to account for noise that may have been produced by the recording technology or by the audience. I used threshold values of 0.33% of the number of annotations for each speech. Empirically, I found that this threshold kept as much of the vocal range as possible while eliminating most of the frequencies caused by the audience.

Results and Analysis

The raw data for each president for each metric is in Appendix A. I was more interested in how these features were changing over time.

[1] The settings were: Window size: 15; Threshold: 10; Time: 100; Cents: 15; Quality: 0

Table 1 shows how the range changed from speech to speech for each president sampled.

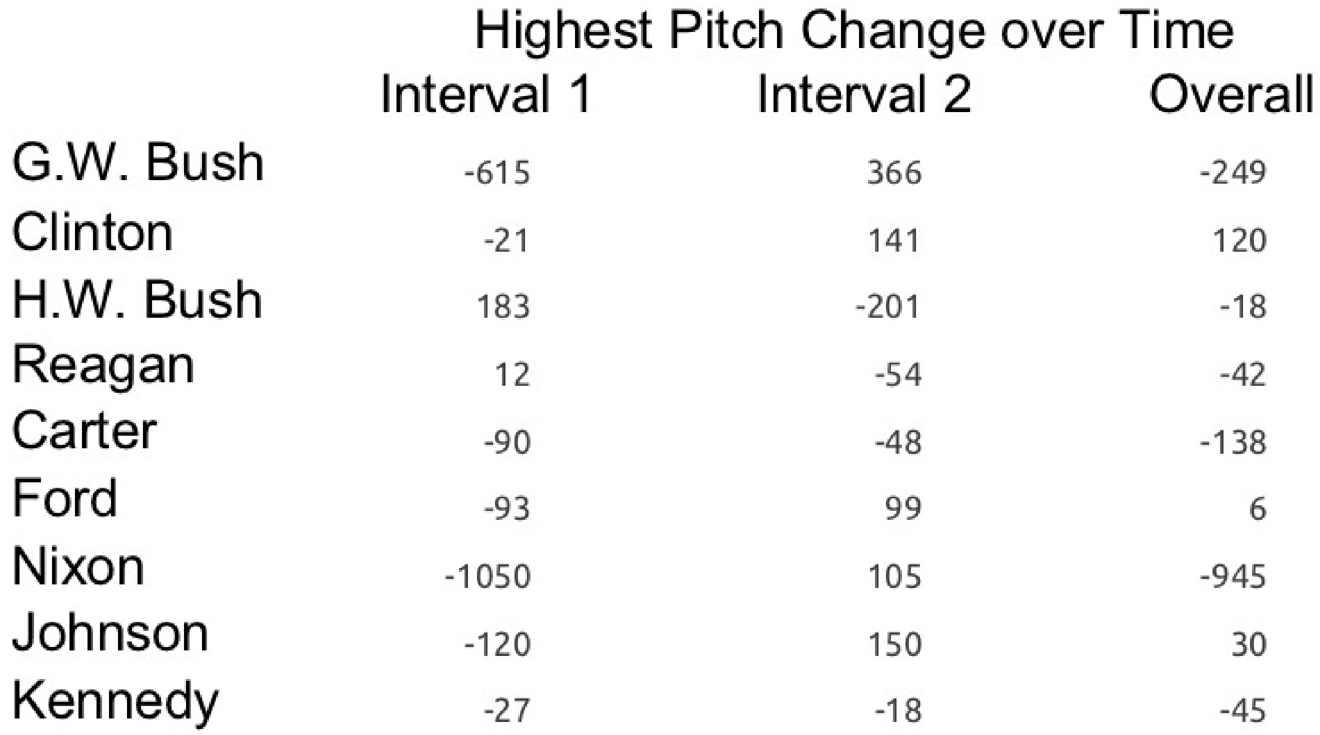


Range Change over Time

	Interval 1	Interval 2	Overall
G.W. Bush	-66	-84	-150
Clinton	-90	141	51
H.W. Bush	249	-396	-147
Reagan	48	-51	-3
Carter	81	9	90
Ford	-45	93	48
Nixon	-792	-87	-879
Johnson	243	-81	162
Kennedy	-24	21	-3

Table 1: Change over time of Pitch Range (in cents above 8.176 Hz)

Between the 1st and 2nd speech, only 44% of presidents sampled increased the size of their range. Going from the 2nd to the 3rd speech, again 44% of presidents sampled increased the size of their range. Over the course of their entire presidency (between their 1st and 3rd speech), again only 44% of them increased the size of their range. As consistent as these numbers are, I believe that they show that there is no strong trend showing that the size of the pitch range of presidents increases over the course of their first term in office.



Highest Pitch Change over Time

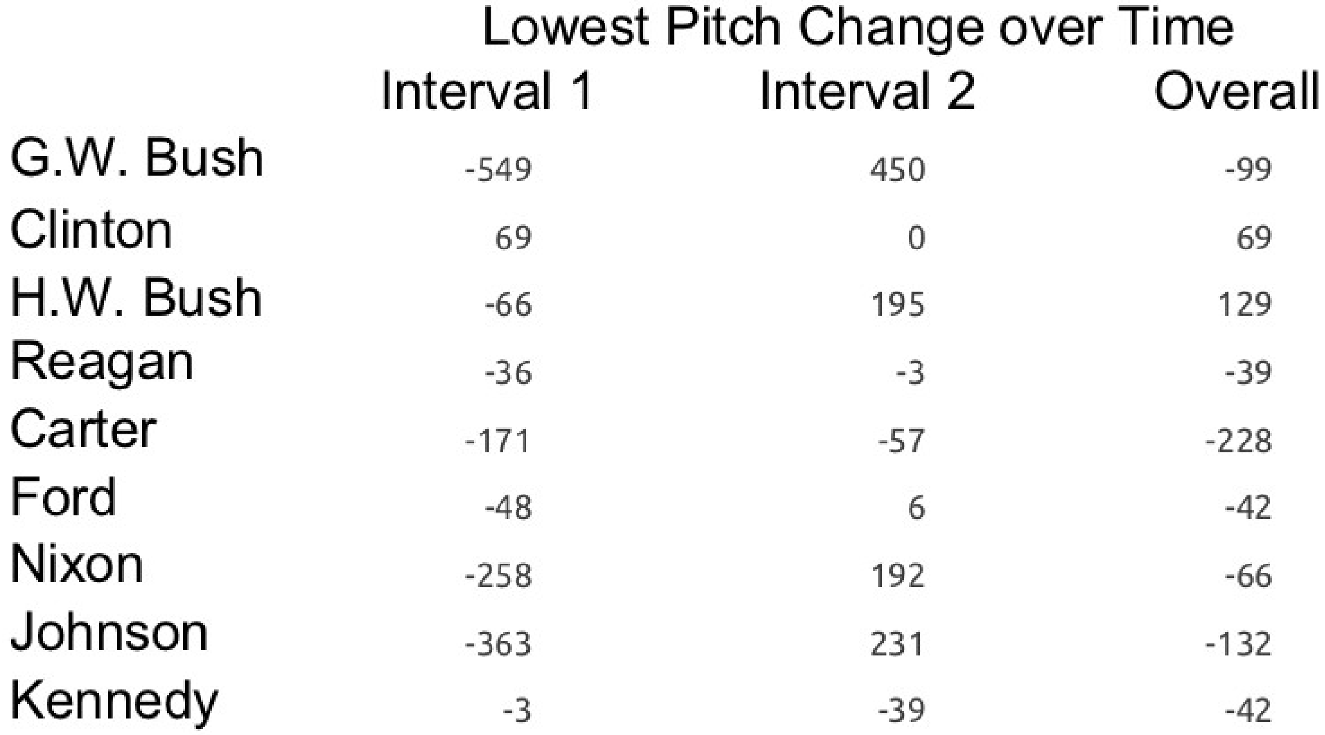
	Interval 1	Interval 2	Overall
G.W. Bush	-615	366	-249
Clinton	-21	141	120
H.W. Bush	183	-201	-18
Reagan	12	-54	-42
Carter	-90	-48	-138
Ford	-93	99	6
Nixon	-1050	105	-945
Johnson	-120	150	30
Kennedy	-27	-18	-45

Table 2: Change over time of Highest Pitch used (in cents above 8.176 Hz)

Tables 2 shows how the highest pitch used in each speech varied over time from speech to speech.

Between the 1st and 2nd speech, 78% of presidents sampled decreased their highest pitch used. Going from the 2nd to the 3rd speech, only 44% of presidents sampled decreased the size of their highest pitch used. Over the course of their entire presidency (between their 1st and 3rd speech), 56% of them decreased their highest pitch used. I believe that these numbers show that there is no strong trend showing that the highest pitch used of presidents decreases over the course of their first term in office, although it is interesting to notice patterns emerging over smaller lengths of time.

Table 3 shows how the lowest pitch used in each speech varied over time from speech to speech



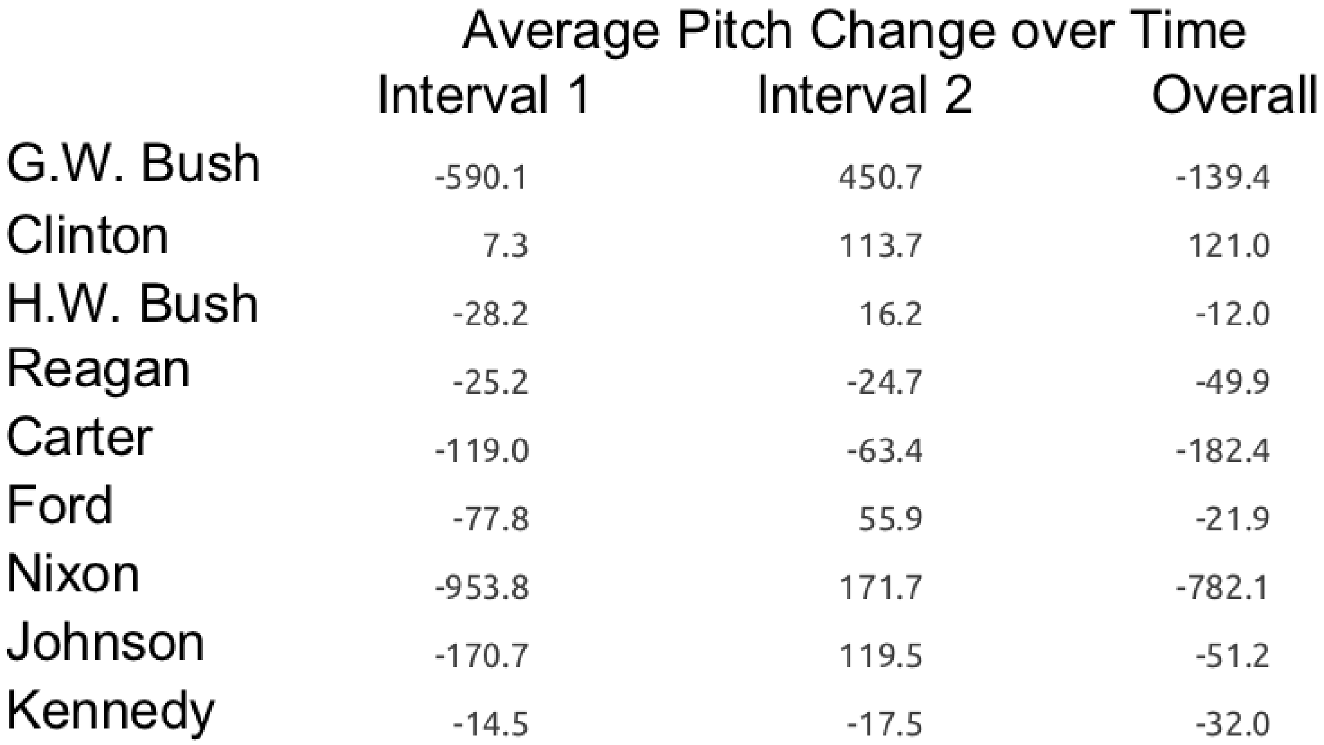
Lowest Pitch Change over Time

	Interval 1	Interval 2	Overall
G.W. Bush	-549	450	-99
Clinton	69	0	69
H.W. Bush	-66	195	129
Reagan	-36	-3	-39
Carter	-171	-57	-228
Ford	-48	6	-42
Nixon	-258	192	-66
Johnson	-363	231	-132
Kennedy	-3	-39	-42

Table 3: Change over time of average Pitch used (in cents above 8.176 Hz)

Between the 1st and 2nd speech, 89% of presidents sampled decreased their lowest pitch used. Going from the 2nd to the 3rd speech, only 44% of presidents sampled did not increase the size of their lowest pitch used. Over the course of their entire presidency (between their 1st and 3rd speech), 78% of them decreased their lowest pitch used. I believe that these numbers might reveal a trend towards the range of pitches used decreasing in frequency over time, or at least that the lower pitches get lower.

Table 4 shows how the average pitch used in each speech varied over time from speech to speech.



Average Pitch Change over Time

	Interval 1	Interval 2	Overall
G.W. Bush	-590.1	450.7	-139.4
Clinton	7.3	113.7	121.0
H.W. Bush	-28.2	16.2	-12.0
Reagan	-25.2	-24.7	-49.9
Carter	-119.0	-63.4	-182.4
Ford	-77.8	55.9	-21.9
Nixon	-953.8	171.7	-782.1
Johnson	-170.7	119.5	-51.2
Kennedy	-14.5	-17.5	-32.0

Table 4: Change over time of Average Pitch used (in cents above 8.176)

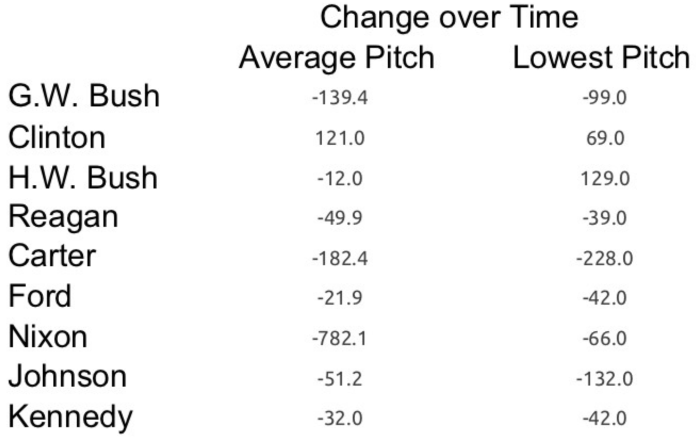
Between the 1st and 2nd speech, 89% of presidents sampled decreased their average pitch used. Going from the 2nd to the 3rd speech, only 33% of presidents sampled decreased the size of their average pitch used. Over the course of their entire presidency (between their 1st and 3rd speech), 89% of them decreased their average pitch used. I believe that these numbers (especially in light of the data for the lowest pitches used) suggest that the pitch range of presidents decreases in frequency over the course of their first term in office.

Previous and Future Work

While no one has attempted to use automatic pitch annotation software to analyze the speech patterns of various presidents before, there has been other work into analyzing speech pitch patterns and also speech synthesis. Atal et. al. used an alternative representation of a speech and attempted to synthesize speech by analyzing the waveforms of other speeches[1]. Using Tarsos, it might be possible to recreate speeches through careful selection and application of the Scala files that are generated. Another next step could be to examine the pitch contours within a particular speech and see how those pitch contours vary with time. It would also be interesting to examine the intentional use of pauses within each speech over time. One could also determine if there is a correlation between the rate of speaking and the average pitch or pitch range used.

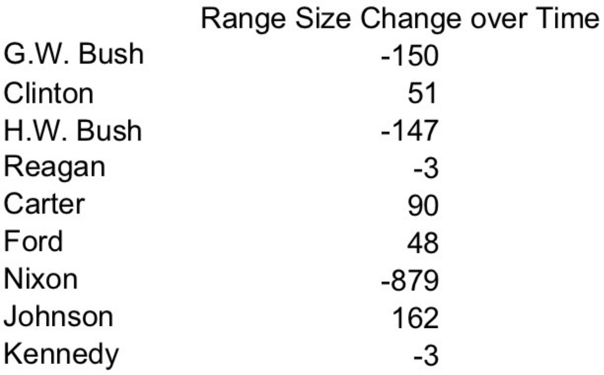
Conclusions

I made two hypotheses. First was that the pitch range of a president increases in size over the course of his first term in office. I believe that the data shows that there is no strong trend to suggest this. Second, I hypothesized that the pitch range of a president decreases in frequency over the course of his first term in office. I believe that the data does, in fact, support this hypothesis.



	Change over Time	
	Average Pitch	Lowest Pitch
G.W. Bush	-139.4	-99.0
Clinton	121.0	69.0
H.W. Bush	-12.0	129.0
Reagan	-49.9	-39.0
Carter	-182.4	-228.0
Ford	-21.9	-42.0
Nixon	-782.1	-66.0
Johnson	-51.2	-132.0
Kennedy	-32.0	-42.0

Table 5: Summarized results supporting the hypothesis that the pitch frequency of the range decreases



	Range Size Change over Time
G.W. Bush	-150
Clinton	51
H.W. Bush	-147
Reagan	-3
Carter	90
Ford	48
Nixon	-879
Johnson	162
Kennedy	-3

Table 6: Summarized results showing that there is no strong trend of increasing the size of the range over the first term of presidency

References

[1] Atal, B. S. "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave." *The Journal of the Acoustical Society of America* 47.1A (1970): 637-55. Print.

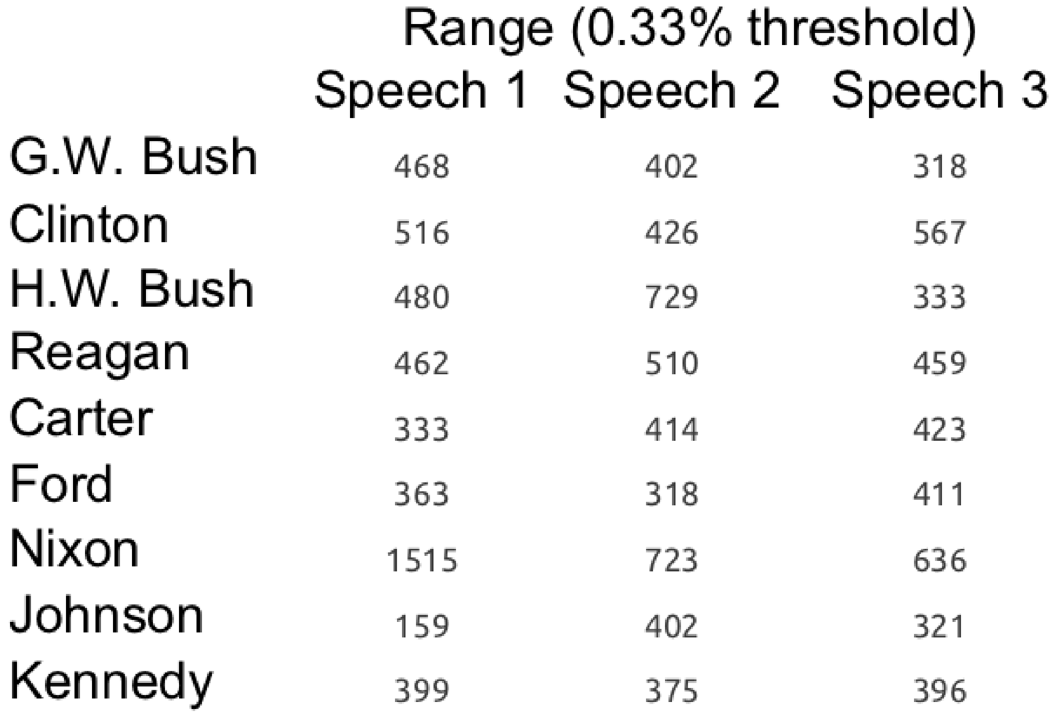
[2] *Presidential Speech Archive*. The Miller Center at the University of Virginia. Web. 9 May 2012. <http://millercenter.org/president/speeches>.

[3] Six, Joren, and Olmo Cornelis. *Tarsos - a Platform to Explore Pitch Scales in Non-Western and Western Music*. *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*. International Society for Music Information Retrieval. Print.

[4] Op De Coul, Manuel. "Scala Scale File Format." *Scala Scale File (.scl) Format*. 2001. Web. 9 May 2012. <http://www.huygens-fokker.org/scala/scl_format.html>.

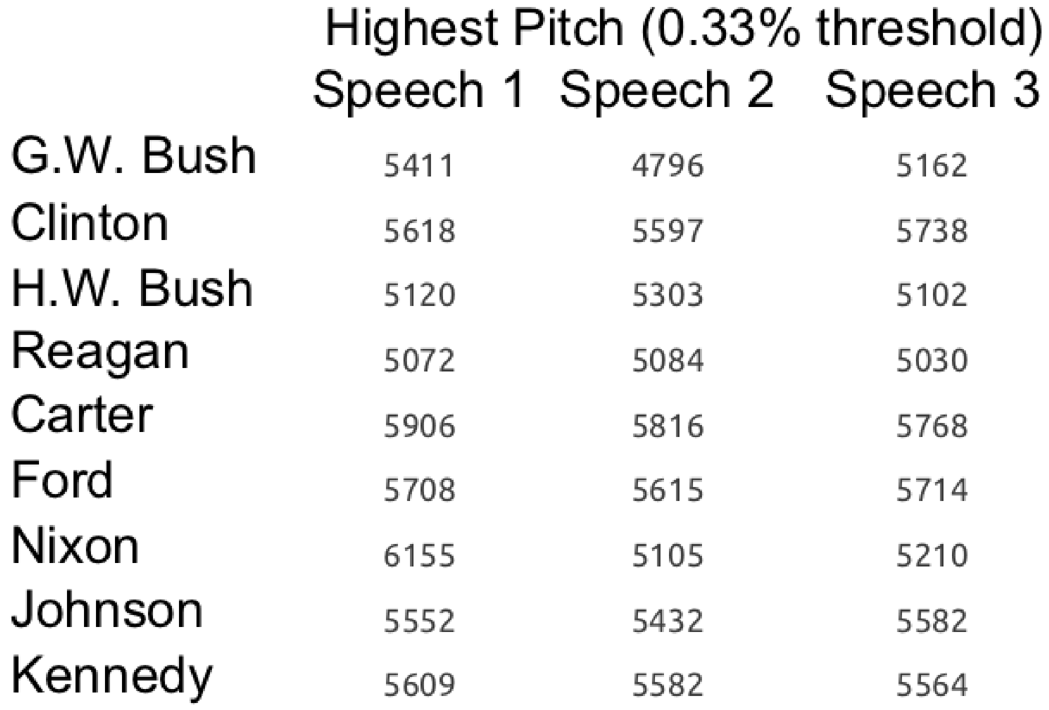
[5] Cuthbert, Michael Scott and Christopher Ariza, "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data," *Proceedings of the International Symposium on Music Information Retrieval 11* (2010), pp. 637–42.

Appendix A – Raw Data



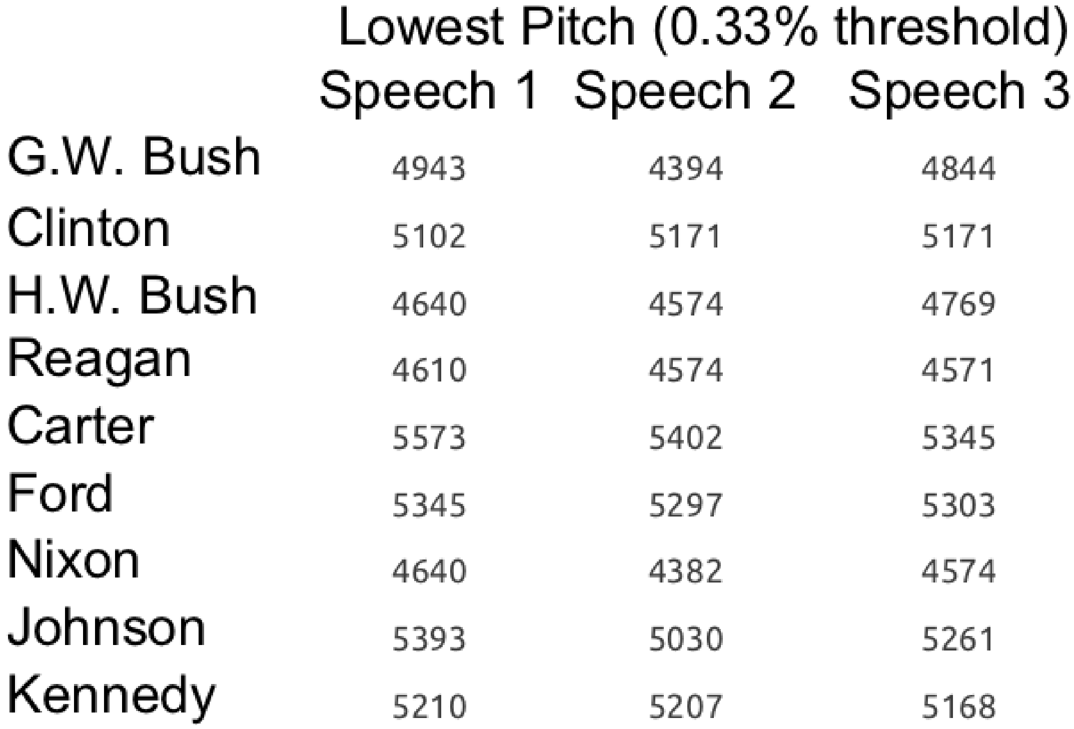
	Range (0.33% threshold)		
	Speech 1	Speech 2	Speech 3
G.W. Bush	468	402	318
Clinton	516	426	567
H.W. Bush	480	729	333
Reagan	462	510	459
Carter	333	414	423
Ford	363	318	411
Nixon	1515	723	636
Johnson	159	402	321
Kennedy	399	375	396

Table 7: Range of each Speech (in cents above 8.176 Hz). Only accepted pitches with more annotations than .0033 times the total number of annotations



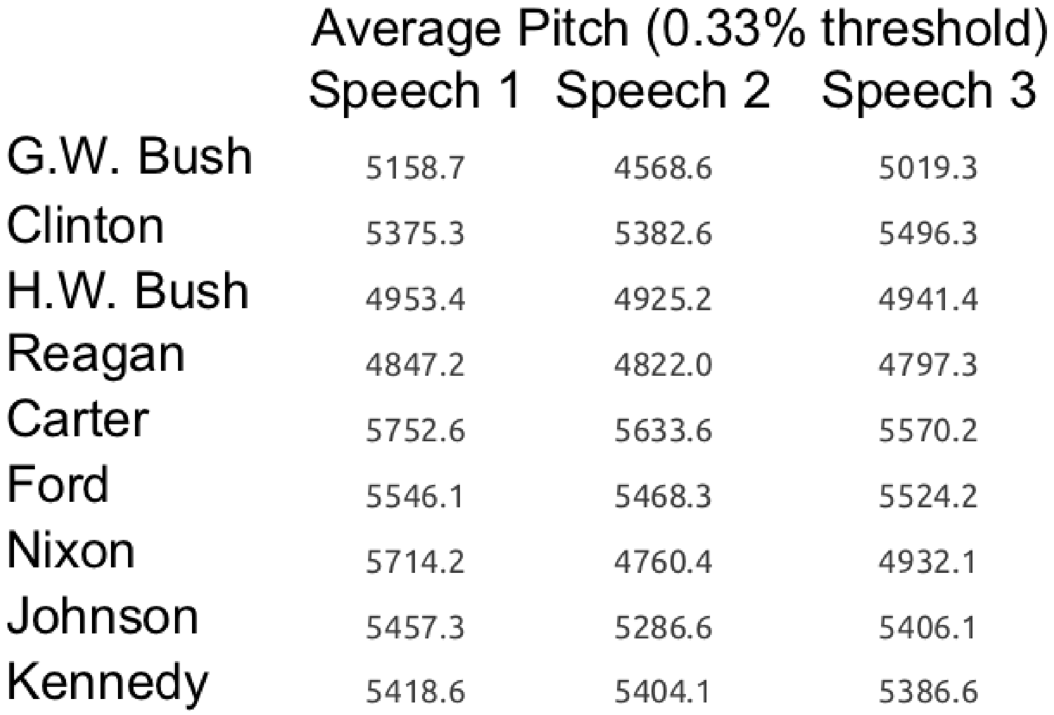
	Highest Pitch (0.33% threshold)		
	Speech 1	Speech 2	Speech 3
G.W. Bush	5411	4796	5162
Clinton	5618	5597	5738
H.W. Bush	5120	5303	5102
Reagan	5072	5084	5030
Carter	5906	5816	5768
Ford	5708	5615	5714
Nixon	6155	5105	5210
Johnson	5552	5432	5582
Kennedy	5609	5582	5564

Table 8: Highest Pitch Frequency per Speech (in cents above 8.176 Hz)



	Lowest Pitch (0.33% threshold)		
	Speech 1	Speech 2	Speech 3
G.W. Bush	4943	4394	4844
Clinton	5102	5171	5171
H.W. Bush	4640	4574	4769
Reagan	4610	4574	4571
Carter	5573	5402	5345
Ford	5345	5297	5303
Nixon	4640	4382	4574
Johnson	5393	5030	5261
Kennedy	5210	5207	5168

Table 9: Lowest Pitch Frequency used per Speech (in cents above 8.176 Hz)



	Average Pitch (0.33% threshold)		
	Speech 1	Speech 2	Speech 3
G.W. Bush	5158.7	4568.6	5019.3
Clinton	5375.3	5382.6	5496.3
H.W. Bush	4953.4	4925.2	4941.4
Reagan	4847.2	4822.0	4797.3
Carter	5752.6	5633.6	5570.2
Ford	5546.1	5468.3	5524.2
Nixon	5714.2	4760.4	4932.1
Johnson	5457.3	5286.6	5406.1
Kennedy	5418.6	5404.1	5386.6

Table 10: Average Pitch Frequency used per Speech (in cents above 8.176 Hz)

Appendix B – Analysis Code

```
from sys import stdout
from music21 import *
import numpy

defaultThreshold = 10

presidentToSpeechDates = {
	'hwBush': ['1990_0131', '1991_0129', '1992_0128'],
	'bush': ['2002_0129', '2003_0128', '2004_0120'],
	'carter': ['1978_0119','1979_0123','1980_0123'],
	'clinton': ['1994_0125','1995_0124','1996_0123'],
	'ford': ['1975_0115','1976_0119','1977_0112'],
	'johnson': ['1964_0108','1965_0104','1966_0112'],
	'kennedy': ['1961_0130','1962_0111','1963_0114'],
	'nixon': ['1970_0122','1971_0122','1972_0120'],
	'reagan': ['1982_0126','1983_0125','1984_0125']
}

presidentToFolderName = {}
for president in presidentToSpeechDates.keys():
	presidentToFolderName[president] = president
presidentToFolderName['hwBush'] = 'bush'

paperKeys = ['average_pitch_default', 'highest_pitch_default', \
	'lowest_pitch_default', 'range_default']

def getAverageInterval(presidentScala):
	'''
	Returns the average interval in cents of the scala
	'''
	return numpy.average([i.cents for i in presidentScala.getIntervalSequence()\
		])

def getAverageCentsAboveTonic(presidentScala):
	'''
	Returns the average cents above tonic for the scala

	Could be interpereted as the middle of the scale
	'''
	return numpy.average(presidentScala.getCentsAboveTonic())

def getMostCommonPitchClass(annotationsToCents):
	'''
	Returns the frequency in cents of the most common pitch class
	'''
	return annotationsToCents[max(annotationsToCents.keys())]
```

```
def getMostCommonPitch(annotationsToCents):
    '''
    Returns the frequency in cents of the most common pitch
    '''
    return annotationsToCents[max(annotationsToCents.keys())]

def getAveragePitch(centsToAnnotations, threshold):
    '''
    Returns the average pitch above a certain threshold
        weighted by it's occurence
    '''
    validPitches = [cent for cent in centsToAnnotations.items() if cent[1] > \
        threshold]
    pitches = []
    for (pitch, annotations) in validPitches:
        pitches += [pitch] * annotations
    return numpy.average(pitches)

def getHistograms(president, speechDate, directory=\
    '/home/the8ball/Documents/term8/21M.269/final/' ):
    '''
    Returns 4 histograms:
        centsToAnnotations for pitch classes
        centsToAnnotations for pitches
        annotationsToCents for pitch classes
        annotationsToCents for pitches
    '''
    presidentPath = directory + president + '/' + 'spe_' + speechDate + '_' +\
        president
    myFiles = [open(presidentPath + '_pitch_histogram.csv', 'r'), \
        open(presidentPath + '_pitch_class_histogram.csv', 'r')]
    centsToAnnotations = {}
    centsToAnnotationsClasses = {}
    annotationsToCents = {}
    annotationsToCentsClasses = {}
    for line in myFiles[0]:
        (cents, annotations) = line.split(';')
        try:
            cents = float(cents)
            annotations = int(annotations)
            centsToAnnotations[cents] = annotations
            annotationsToCents[annotations] = cents
        except:
            #Key or Value can't be converted to a number. Probably means we've
            #    reached the header
            if cents != 'Bin (cents)':
                print "couldn't parse:", cents, annotations
            pass
    for line in myFiles[1]:
        (cents, annotations) = line.split(';')
        try:
            cents = float(cents)
            annotations = int(annotations)
            centsToAnnotationsClasses[cents] = annotations
            annotationsToCentsClasses[annotations] = cents
        except:
            #Key or Value can't be converted to a number. Probably means we've
            #        reached the header
            if cents != 'Bin (cents)':
                print "couldn't parse:", cents, annotations
    return (centsToAnnotationsClasses, centsToAnnotations, \
        annotationsToCentsClasses, annotationsToCents)

def getPresidentialScala(president, speechDate, directory=\
    '/home/the8ball/Documents/term8/21M.269/final/' ):
    '''
    Returns a ScalaStorage object for this speech for this president
    '''
    presidentPath = directory + president + '/' + 'spe_' + speechDate + '_' + \
        president
    return scala.parse(presidentPath + '.scl')

def generatePresidentialHistograms(threshold=defaultThreshold, \
    printProgress=True):
    '''
    Returns a dictionary from president last names to information regarding
        their first 3 state of the union addresses
    '''
    #Used for printing progress
    progress = 0
    progressPerPresident = 1.0 / (len(presidentToFolderName.keys()) * 1.0)

    presidentialAnalysis = {}
    for (president, presidentFolderName) in presidentToFolderName.items():

        if printProgress:
            bars = int(progress * 78)
            spaces = 78 - bars
            stdout.write('|' + '='*bars +  ' '*spaces + '|')
            stdout.flush()

        speechDates = presidentToSpeechDates[president]
        presidentialAnalysis[president] = {}
        progressPerSpeech = progressPerPresident * (1.0 / \
            (len(speechDates) * 1.0))
        for i in xrange(len(speechDates)):
            speechDate = speechDates[i]
            speechName = 'state_of_the_union' + str(i+1)

            #See the getHistograms function for the order of the histograms
            histograms = getHistograms(presidentFolderName, speechDate)
            thisScala = getPresidentialScala(presidentFolderName, speechDate)

            presidentialAnalysis[president][speechName] = {
                'date': speechDate,
                'scala': thisScala,
                'cents_to_annotations_classes': histograms[0],
                'cents_to_annotations': histograms[1],
                'annotations_to_cents_classes': histograms[2],
                'annotations_to_cents': histograms[3],
                'average_interval': getAverageInterval(thisScala),
                'average_cents_above_tonic': \
                    getAverageCentsAboveTonic(thisScala),
                'most_common_pitch_class': \
                    getMostCommonPitchClass(histograms[2]),
                'most_common_pitch': getMostCommonPitch(histograms[3]),
                'average_pitch': \
                    getAveragePitch(histograms[1], threshold=threshold),
                'average_pitch_class': \
                    getAveragePitch(histograms[0], threshold=threshold)
            }

            progress += progressPerSpeech

            if printProgress:
                bars = int(progress * 78)
                spaces = 78 - bars
                stdout.write('|' + '='*bars +  ' '*spaces + '|')
                stdout.flush()

    if printProgress:
        stdout.write('|' + '='*78 + '|')
        stdout.flush()
        print 'Completed'

    return presidentialAnalysis

def picklePresidentialInfo(filename = 'analysis2.pkl', printProgress=True, \
    directory='/home/the8ball/Documents/term8/21M.269/final/' ):
    import pickle
    myFile = open(directory + str(filename), 'w')
    pickle.dump(generatePresidentialHistograms(printProgress = printProgress), \
        myFile)
    print 'Dumped succesfully to', filename

def unpicklePresidentialInfo(filename = 'analysis.pkl'):
    import pickle
    globals()['allPresidents'] = pickle.load(open(filename, 'r'))
    print 'Created allPresidents'
    for (thisPresident, thisHist) in allPresidents.items():
        globals()[str(thisPresident)] = President(thisHist)
        print 'Created %s' % str(thisPresident)

def gatherAllDataForAllPresidents():
    '''
    Returns all the data for all the presidents sorted by key
    '''
    if 'allPresidents' not in globals():
        unpicklePresidentialInfo()

    result = {}
    allDataKeys = []
    for thisPresident in allPresidents.keys():
        if allDataKeys == []:
            allDataKeys = globals()[thisPresident].getDataKeys()
        for thisKey in allDataKeys:
            if thisKey not in result:
                    result[thisKey] = {}
            try:
                    result[thisKey][thisPresident] = globals()[thisPresident].\
                            getDataFromSpeeches(thisKey)
            except KeyError:
                    print "President %s doesn't have the key %s" % (thisPresident, \
                            thisKey)
    return result

def displayDictNicely(thisDict, indentation=0):
    '''
    Displays each item of this dictionary on a new line
    '''
    for (key, value) in thisDict.items():
        if isinstance(value, dict):
            print '\t'*indentation + str(key)
            displayDictNicely(value, indentation=indentation+1)
        else:
            print '\t'*indentation, key, value
```

```
def displayChangeOverTime(hist, keys, indentation=0, showOverallChange=True, \
    showOriginalValues=False):
    '''
    Displays the difference between speeches for a given key
    Dict should be sorted by data key first, and then by president
    '''
    if not isinstance(keys, list):
        keys = [keys]

    for dataKey in keys:
        print '\t' * indentation + dataKey
        for (key, value) in hist[dataKey].items():
            values = value.values()
            data = {key:{'Change Over Time':[values[1] - values[0], values[2] -\
                values[1]]}}
            if showOverallChange:
                data[key]['Overall Change'] = values[-1] - values[0]
            if showOriginalValues:
                data[key]['Original Values'] = values
            displayDictNicely(data, indentation=indentation+1)
        print

class President(object):
    '''
    Allows for easy analysis of presidential histograms
    '''

    def     __init__(self, presidentialHistogram):
        self.hist = presidentialHistogram
        self.histogramNames = ['cents_to_annotations_classes', \
            'cents_to_annotations', 'annotations_to_cents_classes', \
            'annotations_to_cents']
        self.speechPrefix = 'state_of_the_union'
        self.getHighestPitch(default=True)
        self.getLowestPitch(default=True)
        self.getRange(default=True)
        self.getAveragePitch(default=True)


    def getSpeechData(self, speechNumber, showHist=False):
        '''
        Shows all of the analysed data for the given speech but the histograms

        only show the histograms if showHist=True
        '''
        result = dict(self.hist[self.speechPrefix + str(speechNumber)])
        if not showHist:
            for thisName in self.histogramNames:
                del result[thisName]
        return result

    def getDataKeys(self):
        '''
        Return the valid data keys for this president
        '''
        return self.hist['state_of_the_union1'].keys()

    def getDataFromSpeeches(self, dataKey, speeches=[1,2,3]):
        '''
        Returns the specified data key for each of the speeches in the list
            'speeches'
        '''
        result = {}
        for speechNum in speeches:
            try:
                result[speechNum] = self.hist[self.speechPrefix + \
                    str(speechNum)][dataKey]
            except KeyError:
                print "The speech %s doesn't have the key %s" % \
                    (self.speechPrefix + str(speechNum), dataKey)
        return result

    def getDataFromSpeech(self, dataKey, speechNum):
        '''
        Returns the specified data key for the specified speech
        '''
        return self.getDataFromSpeeches(dataKey=dataKey, speeches=[speechNum])

    def getAllData(self, showHist=False):
        '''
        Gets all data for all speeches excluding the histograms

        unless showHist is True
        '''
        result = dict(self.hist)
        if not showHist:
            for i in xrange(1,4):
                for thisName in self.histogramNames:
                    del result[self.speechPrefix + str(i)][thisName]
        return result

    def getAllDataByKeys(self, showHist=False):
        '''
        Gets all data for all speeches excluding the histograms
            unless showHist is True

        Returns an object sorted by the Data key
        '''
        result = {}
        for thisKey in self.getDataKeys():
            if thisKey in self.histogramNames and not showHist:
                continue
            result[thisKey] = {}
            for i in xrange(1,4):
                try:
                    result[thisKey][i] = self.hist[self.speechPrefix + \
                        str(i)][thisKey]
                except KeyError:
                    print "The speech %s doesn't have the key %s" % \
                        (self.speechPrefix + str(speechNum), dataKey)
        return result

    def getHighestPitch(self, speeches=[1,2,3], threshold=None, \
        thresholdPercentage=0.33, default=False):
        '''
        Returns the value of the highest pitch used (in cents)

        Must occur more than threshold times
        '''
        if not isinstance(speeches, list):
            speeches = [speeches]

        baseName = 'highest_pitch_'
        if default:
            baseName = baseName + 'default'

        results = {}
        for speech in speeches:
            thisName = baseName
            if threshold==None and thresholdPercentage==None:
                thisThreshold = int(self.getNumAnnotations(speech) / 200.0)
                if not default: thisName += str(thisThreshold)
            elif threshold==None:
                thisThreshold = int(self.getNumAnnotations(speech) * \
                    thresholdPercentage / 100.0)
                if not default: thisName += str(thresholdPercentage)
            else:
                thisThreshold = threshold
                if not default: thisName += str(thisThreshold)
            if thisName not in self.hist[self.speechPrefix + str(speech)]:
                validPitches = [pair[0] for pair in self.hist[self.speechPrefix\
                    + str(speech)]['cents_to_annotations'].items() if pair[1] >\
                    thisThreshold]
                self.hist[self.speechPrefix + str(speech)][thisName] = \
                    max(validPitches)
            results[speech] = self.hist[self.speechPrefix + str(speech)]\
                [thisName]
        return results

    def getLowestPitch(self, speeches=[1,2,3], threshold=None, \
        thresholdPercentage=0.33, default=False):
        '''
        Returns the value of the lowest pitch used (in cents)

        Must occur more than threshold times
        '''
        if not isinstance(speeches, list):
            speeches = [speeches]

        baseName = 'lowest_pitch_'
        if default:
            baseName = baseName + 'default'

        results = {}
        for speech in speeches:
            thisName = baseName
            if threshold==None and thresholdPercentage==None:
                thisThreshold = int(self.getNumAnnotations(speech) / 200.0)
                if not default: thisName += str(thisThreshold)
            elif threshold==None:
                thisThreshold = int(self.getNumAnnotations(speech) * \
                    thresholdPercentage / 100.0)
                if not default: thisName += str(thresholdPercentage)
            else:
                thisThreshold = threshold
                if not default: thisName += str(thisThreshold)

            if thisName not in self.hist[self.speechPrefix + str(speech)]:
                validPitches = [pair[0] for pair in self.hist[self.speechPrefix\
                    + str(speech)]['cents_to_annotations'].items() if pair[1] >\
                    thisThreshold]
                self.hist[self.speechPrefix + str(speech)][thisName] = \
                    min(validPitches)
            results[speech] = self.hist[self.speechPrefix + str(speech)]\
                [thisName]
        return results
```

```
def getRange(self, speeches=[1,2,3], threshold=None, \
    thresholdPercentage=0.33, default=False):
    '''
    Returns the distance in cents between the highest used pitch and the lowest one
    '''
    if not isinstance(speeches, list):
        speeches = [speeches]

    baseName = 'range_'
    if default:
        baseName = baseName + 'default'

    results = {}
    for speech in speeches:
        thisName = baseName
        if threshold==None and thresholdPercentage==None:
            thisThreshold = int(self.getNumAnnotations(speech) / 200.0)
            if not default: thisName += str(thisThreshold)
        elif threshold==None:
            thisThreshold = int(self.getNumAnnotations(speech) * \
                thresholdPercentage / 100.0)
            if not default: thisName += str(thresholdPercentage)
        else:
            thisThreshold = threshold
            if not default: thisName += str(thisThreshold)

        if thisName not in self.hist[self.speechPrefix + str(speech)]:
            validPitches = [pair[0] for pair in self.hist[self.speechPrefix\
                + str(speech)]['cents_to_annotations'].items() if pair[1] >\
                thisThreshold]
            self.hist[self.speechPrefix + str(speech)][thisName] = \
                max(validPitches) - min(validPitches)
        results[speech] = self.hist[self.speechPrefix + str(speech)]\
            [thisName]
    return results

def getNumAnnotations(self, speechNum):
    '''
    returns the total number of annotations for a given speech
    '''
    return sum(self.hist[self.speechPrefix + str(speechNum)]\
        ['cents_to_annotations'].values())

def getAverageNumAnnotations(self):
    '''
    returns the average number of annotations per speech
```

```
        '''
        return numpy.average([self.getNumAnnotations(i) for i in xrange(1,4)])

    def getAveragePitch(self, speeches=[1,2,3], threshold=None, \
        thresholdPercentage=0.33, default=False):
        '''
        Returns the average pitch above a certain threshold
            weighted by it's occurence
        '''
        if not isinstance(speeches, list):
            speeches = [speeches]

        baseName = 'average_pitch_'
        if default:
            baseName = baseName + 'default'

        results = {}

        for speech in speeches:
            thisName = baseName
            if threshold==None and thresholdPercentage==None:
                thisThreshold = int(self.getNumAnnotations(speech) / 200.0)
                if not default: thisName += str(thisThreshold)
            elif threshold==None:
                thisThreshold = int(self.getNumAnnotations(speech) * \
                    thresholdPercentage / 100.0)
                if not default: thisName += str(thresholdPercentage)
            else:
                thisThreshold = threshold
                if not default: thisName += str(thisThreshold)
            validPitches = [cent for cent in self.hist[self.speechPrefix + \
                str(speech)]['cents_to_annotations'].items() if cent[1] > \
                thisThreshold]
            pitches = []
            for (thisPitch, annotations) in validPitches:
                pitches += [thisPitch] * annotations

            if thisName not in self.hist[self.speechPrefix + str(speech)]:
                self.hist[self.speechPrefix + str(speech)][thisName] = \
                    numpy.average(pitches)
            results[speech] = self.hist[self.speechPrefix + str(speech)]\
                [thisName]
        return results
```