Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

# Preparing Acoustic Pitch Data for Computational Analysis and Presentation

Kellen Parker van Dam
Chair for Multilingual Computational Linguistics
University of Passau

Pitch plays an important role in many linguistic systems. It is the primary set of features which determine vowel quality distinctions as well as forming the basis for intonation and contrastive tone systems. Unfortunately, much of the literature has relied on approaches to presenting and analysing pitch data that can result in a lack of data transparency, reproducibility, and analytical robustness. These issues are easily solved through the selection of a more appropriate scale for pitch values. This study presents the issues with using raw pitch data as Hertz values some historical efforts to resolve these issues, and two more appropriate solutions than some of the more widely used systems, with a way to easily calculate these alternative systems in a short Python script.

## 1 Introduction

A common issue seen in the literature on pitch is the representation of raw pitch data in Hertz values (hereafter Hz) as well as the use of Hz as the basis for analysing features. This is especially important when comparing between languages or speakers, where a certain degree of normalisation is likely necessary. There are a number of reasons why using raw Hz values is something to be avoided in many cases. In this study we will go over some of the important features of pitch as a meaningful linguistic marker, as well as past efforts to resolve issues around using Hz by coming up with a more psychoacoustically relevant scale. We will also discuss why it is important to always still include raw data in Hz for the sake of experimental reproducibility, but also bring into question the appropriateness of using Hz as the basis for making arguments around or descriptions of some of the features for which pitch is so fundamental.

These issues around selecting appropriate scales when dealing with pitch are a common concern when dealing with linguistic description for systems such as intonational patterns, lexical tone systems, and vowel quality differences or vocalic inventories. Whether looking at cross-linguistic comparisons for typological studies or

*Computer-Assisted Language Comparison in Practice*
Volume 7, Number 2
URL: https://calc.hypotheses.org/7390 DOI: 10.15475/calcip.2024.2.4

Published on 07/10/2024

language-internal systems, considerations around the handling of pitch data are of great importance to ensure good science. It is still common for modern publications to present data solely based on pitch, or worse yet relying on scales developed on unsound methodologies. Thus we will look at a comparison of different scales used frequently in the literature, explain why Semitones or log-z Scores are preferable, and then look at how to calculate these programmatically from pitch data.

## 2 Psychoacoustic Pitch Perception

### 2.1 Hz as the Ground Truth

When investigating any linguistic feature reliant on pitch, values in Hz are always the starting point. Vowel charts are simply visualisations of relative pitch values of vowel formants, at least for F1 and F2, which are underlyingly Hz values. Tone and intonation data are pitch values of the fundamental frequency, otherwise known as F0. All information which can be gleaned from a spectrogram is made up of pitch values in Hz, regardless of whether we're looking at the time domain waveform or frequency domain spectrogram. It is the raw data, reflecting the physical properties of sound waves moving through the air.

For the sake of openness of data, Hz should always be included as supplementary material. It is the bare minimum for being able to reproduce results. However, using Hz values for the representations of pitch data is almost never appropriate. Nor is it usually appropriate for analysis of the data, especially when we are dealing with more than one speaker.

### 2.2 Over-Reliance on Hz in Analysis and Presentation

There are a couple of reasons for Hz being less than ideal. The first is that we simply do not interpret pitch psychoacoustically in Hz. Hz is a linear scale, but human pitch perception instead operates logarithmically. The higher the pitch range, the less significant an otherwise equivalent difference in Hz will be. A 5Hz difference may be either readily identifiable to a human listener as a distinct pitch, or entirely imperceptible. The difference is dependent on where that 5Hz change occurs on the full range of frequencies the human ear can detect.

Taking as an example a synthesiser keyboard tuned as 12 equal temperament where the octave is divided into equal parts, a change of 5Hz in the lower octaves could be a full semitone. This is the difference between $E_2$ (at 82.41Hz) to $F_2$ (87.31Hz). As we go higher, the difference doubles with each octave. From $E_5$ to $F_5$, the difference between the notes will instead be almost 40Hz, going from 659.25Hz to 698.45Hz. To our ears and brains, these two differences — 5Hz in one case and 40 in the next — are perceptually the same.

To reliably present pitch data, we must take into account where in the audible pitch range our target pitch occurs. Saying that a certain tone falls some number of Hz over the duration of the tone-bearing unit — that is, the part of the morpheme or word which carries the significant pitch of the toneme or tonal phoneme — is ultimately meaningless if we don't otherwise know more information including the speakers baseline pitch range, where this falls relative to other speakers, and a number of other speaker-specific features such as whether they tend to use their full range or not. When presenting pitch data from a single speaker, Hz might be an easy approach, but it loses much of the important information. Furthermore, to describe pitch values across speakers based on simply averaging Hz values, we are failing to capture what may actually be occurring as it is meaningful to the speaker and listener.

For example, imagine two speakers, who we will call Speaker A and Speaker B. In this exercise, we imagine that Speaker A has a low pitch range relative to the average, and also tends to use less of it than others. Speaker B has a higher than average pitch range, and also makes use of nearly all of it in casual speech. Their Hz values are not comparable. Speaker B will easily have massive changes in Hz from moment to moment compared to Speaker A. Even if the two speakers used most of their range, we'd still see much larger changes with speaker B since they are operating at a much higher average pitch, meaning a perceptually identical fall in tone could still be multiple times greater for Speaker B than A. This is why speaker data should never just be averaged. Instead, we need to normalise the data. In this case normalising means making their ranges and values statistically equivalent in order to get to the underlying pitch representation as it may occur in their brains. There are a number of ways to do this. First, let's look at the problem of scales.

## 3 Efforts To Correct the Shortcomings of Hz

Recognising the mismatch between human pitch perception and Hz, a number of attempts have been made to come up with more psychoacoustically appropriate scales. In this section, the most commonly used scales will be presented and analysed. Note that many of these occur as built-in scales for much of the available audio-processing and speech acoustics software, including Praat (Boersma & Weenink 2024) and Audacity.

### 3.1 Mel Scale

One of the more notable early efforts which is still seen today was that of Stevens et al (1937), revised in Stevens & Volkmann (1940) who attempted to come up with what they referred to as a "sense-distance", being a more accurate representation of human pitch perception. Their experiment consisted of a keyboard with 20 keys. The pitch produced by each key could be independently tuned. Using just five of these keys, the highest and lowest of the five were tuned to a specific value, and then participants were

asked to tune the intermediate steps to be of equal perceptual distances. The experiment was done with three stages each with different overall pitch ranges. The lowest of these three ranges was set to 40-1000batch for the first round of experimentation, then 200-6500Hz for the second, and finally 3000-12000Hz. Only 10 participants took part in the experiment, and their results varied considerably. The experimenters then averaged out the values at which each speaker divided the pitch range, and this became the mel scale, "mel" being short for "melody".

The main issue with the mel scale is that there is not a formula that accurately captures the results. Fant (1968) proposed the following:

$$m = 1000 \cdot \frac{log_e(1 + \frac{x_i}{1000})}{log_e 2}$$

Here $x_i$ refers to the target pitch which we wand to convert to mels and m is the resulting value. The value of 1000 occurs here as 1000 Hz was linked to 1000 mels in the original proposal, and so is in the formula setting the baseline. O'Shaughnessy (1987) presented an alternative formula, producing slightly different values.

$$m = 1127 \cdot log_e(1 + \frac{x_i}{700})$$

$$= 2585 \cdot log_{10}(1 + \frac{x_i}{700})$$

Additional formulae have been attempted as well, each differing slightly. Since the experiment to determine the mel scale had so few participants and the values determined by the participants are not easily captured in a straightforward formula. What's more, the mel scale is logarithmic at higher frequencies, but a linear scale at low frequencies. Since it was intended originally for melodic pitch (hence the "mel" in the name), the pitch ranges used were often considerably higher than those occurring in speech, and at the lower ranges where human speech occurs, it is not any real improvement over just using Hz. Fant (1968) put this change from linear to logarithmic at 1000Hz, well above human pitch ranges for F0. Makhoul & Cosell (1976) instead placed the changeover at 700Hz, again well above a normal F0 value. At 700Hz the scale might be useful for vowel formant frequencies, were it not already based on such a small participant size with such inconsistent values. For these reasons mel as a scale is better off avoided.

### 3.2 Bark Scale

Another commonly used scale is the Bark scale (Zwicker 1961), developed in an attempt to improve upon the mel scale to better capture human pitch perception while still closely

following the mel scale, where one Bark is approximately 100 mels. The Bark scale was based on the idea of critical bands, a proposed explanation of audio processing involving fixed frequency ranges which affect our ability to discern different pitches.

It is an effort to explain auditory masking, where a louder sound may completely mask out our perception of a quieter sound that is close in pitch, even if they would be discernibly different otherwise. Critical band theory is rather coarse, providing a simple model to explain some features of auditory processing. As with the mel scale, multiple formulae have been presented, each with differing results.

The most complex of these is that of Traunmuller (1990) where first an intermediate value is calculated as $r = \frac{25.81\ x_i}{1960 + x_i} - 0.53$. As above, $x_i$ refers to our target value in Hz. Then, if the resulting value, here r, is less than 2, the Bark value is r+0.51(2−r). If the value instead is larger than 20.1, then the Bark value is r+0.22(r−20.1), and if it is in between 2 and 20.1 then the Bark value is simply the original r. The inconsistency in values between different formulae — not to mention the complexity of that in Traunmuller (1990) — should be enough of an indicator that a scale still largely based on mels is again not the answer. While the Bark scale still regularly occurs in descriptions of pitch systems, there are better options.

### 3.3 Equivalent Rectangular Bandwidth (ERB)

The final scale presented here which is still discouraged is the ERB scale. Like the Bark scale, it is based on critical band theory and assumes human acoustic processing is built on a series of band-pass filters. It is also based on the assumption that these critical bands have a physical reality, being equal distances across the cochlear basilar membrane. It was first developed in Greenwood (1961) and built upon in Glasberg and Moore (1990). The formula for ERB from Hz is $ERB(x_i)=24.7\cdot(0.00437\cdot x_i+1)$. ERB like Hz is still a linear scale, however, and so will not be an accurate reflection of human pitch perception. This brings us finally to better alternatives.

## 4 Better Options

As we have seen, many of these scales have considerable shortcomings. There are fortunately better options available to us. Most significant among these are Semitones and logarithmic z-Scores. These two scales are discussed in the following sections, although they are not the only options.

### 4.1 Semitones

Semitones provide a clean solution when investigating pitch data, especially for F0 tonal or intonational data. Semitones can be understood as half-notes on a keyboard tuned to

12 equal temperament, or put another way, 12 equal logarithmic divisions within an octave, where each semitone to its previous step has a ratio of $\sqrt[12]{2}$ to 1. A major benefit of using semitones is that the octave on which it is based appears to be universal across musical traditions: It is described in early Indian texts, as well as early Chinese musicological texts (Chen 1988), and is a common feature in African musical traditions (Kubik 2010).

> It can be seen that for both male and female speakers semitones result in the smallest replication error… Post hoc paired-samples t-tests revealed that most between- scale differences were significant at the $p < 0.01$ level, with the exception of the difference between semitones and ERB-rate for female speakers which was significant only at $p<0.05$. The experiment suggests that a logarithmic (st) or near logarithmic (ERB) scale best models intonational equivalence. (Nolan, 2003)

The calculation of semitones is also straightforward, $s = 12 \cdot log_2(\frac{x_i}{y})$. Here, s is our semitone value, $x_i$ is again our target value in Hz, and y is a pitch value in Hz to set as the baseline value of 0 semitones. Jitwiriyanont (2014) calculated semitones based on the minimum pitch value of the speaker, ensuring that the lowest value would be 0. This has a major drawback in that creaky phonation — a common feature of Southeast Asian tone systems — will result in artificially low pitch values when using tools such as Praat to get pitch. An additional problem is that the lowest pitch across tokens may only occur in a single token, thus your more typical lowest value may be slightly above 0 semitones. Fant (2006) instead suggests 100Hz as the 0 semitone baseline, which helps filter out outliers caused by creak. This is not an ideal solution, since 100Hz will anyway be arbitrary when considering any given speaker; some may have pitch ranges which are often considerably above 100Hz while other speakers may tend to be below this value.

A better option would be instead to calculate the 0 point at the mean pitch value of the speaker's total pitch data across an entire session. It's better to limit to a single session than the entire corpus for that speaker to help accommodate to session-dependent pitch tendencies of the speaker, such as a lower overall mean when tired, or higher when exited, or any other intra-speaker variations from session to session. Setting 0 semitones to the mean Hz value does result in negative semitone values, something which some people may wish to avoid, and a major reason for Jitwiriyanont (2014) using the minimum, but by using the mean pitch value for the speaker, a first step toward cross-speaker normalisation is done; if each speaker's own mean is set to 0 semitones, then speakers with vastly different mean pitches will still have semitone values that are largely comparable. This is only a first step to normalisation, however, since semitones cannot

account for differences in the speakers' ranges, should that be something one also wishes to normalise.

The semitone scale is not without criticisms, but they come from a very different place than criticism of the mel, Bark or ERB scales. Instead, the main criticism of semitones comes from its apparent foundation in Western musical traditions. Therefore among some ethnomusicologists, an alternative is instead preferred, known as the Cent scale. However, in every meaningful way it is identical to semitones, as the cent values are simply the semitone values multiplied by 100 as the name may suggest. Thus 5.52 semitones would be exactly 552 cents. The benefit of semitones over cents in this case is simply one of salience; people are much more likely to be familiar with the concept of a half note over than a $0.005^{th}$ note. The populiser of the system also stated that the difference of a single cent would not be discernible should two notes be played in succession (Ellis 1885). Thus, for most purposes where presenting data from a single speaker, using semitones is recommended. It is salient as a concept and accurately reflects psychoacoustic perception of pitch differences.

Semitones have one major shortcoming which is the inability for cross-speaker normalisation when it comes to pitch ranges. When describing tone systems, vowel formants, intonational tendencies, or anything else where the data are pitch values, multiple speakers should ideally be consulted. And as mentioned, different speakers will have entirely different ranges, whether that be in tone spaces or in how distant their vowel formants may be from each other. As these values differ person to person, a better system than semitones is probably needed. For that, there are logarithmic z-Scores.

### 4.2 Logarithmic z-Scores

Z-Scores are simply the values of data, in our case pitch data, relative to the mean and presented in standard deviations. A z-Score of 1.0 is one standard deviation above the mean, and -1.0 is one standard deviation below the mean. You can calculate z-Scores from Hz directly, but this will still not solve the issue of Hz being a linear scale while human pitch processing is logarithmic. Such z-Scores will also be linear, since they are calculated from linear values. To solve this, we simply need to calculate the z-Scores not from the Hz values, but from something like semitones which are already logarithmic. For this reason, it is recommended that even if you are only using log z-Scores for your data presentation and analysis, converting Hz to semitones first may have some real value, and as we will see below this is easy to do. Furthermore, for the average reader, semitones may still be more salient as a descriptive device than standard deviations from the mean anyway, so this is the process we will take in calculating values below.

The main benefit of log z-Scores is that we not only normalise the pitch values across speakers, but also the pitch ranges. Imagine we are creating a vowel plot for a given language. In addition to pitch differences inherent to the anatomy of the speakers, some

speakers may have a much smaller frequency space for vowel formants. We can easily imagine one speaker who has a higher low-vowel tongue height relative to the high vowels, as compared to other speakers. Were we to simply average all of the speakers' data for a given target vowel, these two major pitch differences (inherent pitch difference plus smaller pitch range) would result in data that didn't really show much of anything meaningful. It might tell us something general, but it will be a less accurate picture. Instead, the vowel plot can take into account both of these inter-speaker differences by plotting not the Hz values of the formants, but the logarithmic z-Scores of the values based on the mean pitch values for F1 and F2. This lets us better normalise between Hz a speaker with a much shallower F1 range and one with considerably greater differences in vowel heights.

The same benefit is achieved when dealing with tone. Speakers who have a smaller tone space will not cause issues for a more language-wide description, as listeners already adjust to individual differences. Logarithmic z-Scores provide an ideal means of normalising any sort of pitch data across speakers, regardless of the type of pitch data we are dealing with.

## 5 Calculating Semitones and Logarithmic z-Scores with Code

Above we have gone through some of the reasons why Hz values are not good for analysis and presentations, some of the efforts to develop something better, and why semitones and logarithmic z-Scores fit that bill. In this section, we will discuss how to take the raw Hz values from software such as Praat and calculate the values we need. For the sake of this discussion, we will be focusing solely on F0 values for lexical tones, but the same can be done with semitone values. We focus on F0 simply to have less complicated data, as we don't need to worry about multiple formants at a time.

While there are different approaches for determining the time step settings in Praat, for our purposes here we will set this to be view-dependent, with 11 steps. The reason for using 11 steps is that it gives us the pitch value at 0% of the duration, and at every 10% after until and including 100% of the total duration. The pitch listing in Praat will then give something like this, with time values in seconds in the first column and Hz values in the second column:

```
Time_s   F0_Hz
144.640339   195.700913
144.657444   193.733349
144.674549   190.020519
144.691654   186.412159
144.708759   184.221198
144.725865   183.369391
144.742970   180.900870
```

Frustratingly, Praat gives the pitch data with three spaces between each column, rather than 2 or 4 or a tab character, so spreadsheet software generally will not detect this as any sort of tab-separated value. A Praat scripting tutorial is somewhat outside the scope of this study, so we will leave it for now and assume we are starting with pitch values in this format, written to individual files via a Praat script. At any rate, we will save a few tokens of a single syllable word in the local ./data folder, and then later loop through those in the Python script, described below.

Once we have the pitch data, we can write a simple Python script which will load the data as a pandas dataframe (McKinney 2010), easily calculate the semitone and logarithmic z-Score values based on the mean value of all samples, and then plot the contours with matplotlib (Hunter 2007). That will be the end goal of this section.

### 5.1 Ensuring Clean Data

The first thing we're going to do is check for bad pitch data. When exporting data from Praat, if the TextGrid window is not correctly aligned, or if you have creaky phonation or some other feature that causes the software to be unable to reliably detect pitch data, you will end up with lines which have `--undefined--` rather than time and pitch data. We obviously do not want that, since it will prevent us from being able to calculate the data correctly. Thus, first ensure that no such lines occur in the data. Once this is done we're able to continue on, assuming that we don't have any bad pitch lists.

### 5.2 Creating the Dataframe and Loading the Data

We're going to use two important packages here. The first is pandas, a Python package for handling dataframes. Pandas is an excellent tool for this purpose and it lets us in a single line of code generate the additional columns we need based on dataframe-wide values, such as an overall mean pitch value. We also need the math package for some functions we will use, and matplotlib for the plot.

```
import numpy as np
from scipy.interpolate import make_interp_spline
import pandas
import math
import matplotlib.pyplot as plt
from pathlib import Path

filelist = list(Path("./data/").glob("*"))
for filename in filelist:
    data = pandas.read_csv(filename, sep="   ", engine="python")
```

We load the file directly as a data frame here, setting the separator to the three spaces that Praat uses. Setting the `engine` flag to `python` suppresses some complaints about an unusual separator, i.e. our three spaces. We can of course have our Praat script instead generate the file with a comma or tab character and not need to set the `engine`, but we'll stick to their default here so that this is able to handle direct saves of Praat data.

### 5.3 Adding New Scale Columns

Now we calculate local semitones. We're calling this "local" because we are not taking into account the entire set of tokens which our analysis may be considering. This is fine, since we're anyway dealing with only one speaker here. If we were to handle multiple speaker's data, we would want to also have a separate dataframe set up to hold every token for each speaker which we can then calculate means by.

```
data["Semitones"] = 12 * data["F0_Hz"].apply(
    lambda x: math.log(x / data["F0_Hz"].mean(), 2))
```

This part of the code is the main reason to use pandas in our case. Assigning the column values with the `.mean()` and `.std()` functions mean we do not need to do any additional calculation, and can handle it in a single line of code. If we wanted to also include cents, pandas again makes this easy as we can create a new column based on the values of another column in the same row.

```
data["Cents"] = round(data["Semitones"] * 100).astype(int)
```

Getting the z-Score is equally simple.

```
data["ZScore"] = (
    data["Semitones"] - data["Semitones"].mean()) / data["Semitones"].std()
```

Note again we are calculating it based on the semitones value, ensuring that we are ending up with a logarithmic z-Score rather than a linear one. We again rely on `.mean()` and `.std()`. Effectively one line of code calculates the semitones based on the mean for the entire dataframe, and one line of code does the same for the log z-Score, with no need to write loops or other data handling methods.

### 5.4 Normalising Durational Data

The other thing we want to do is get a local time value for each sample by subtracting the lowers time value from all time values for that token. This is useful because it allows us to conduct duration-normalised analysis, which can be especially useful when creating simpler visualisations of tone data. We already have the original `Time_s` value directly from the Praat ouput. This is the time in seconds but from the position in the entire audio file from which the data came. We'd like to create a new column where we subtract the

minimum value from each of these values, so that we end up with an eventual time axis that starts at 0.0 seconds. This is in a new column so that we don't lose any of the original data. We'll call this `Time`, leaving `Time_s` as the original values.

```
    data["Time"] = data["Time_s"] - data["Time_s"].min()
```

By creating this new column, it's easier to plot the values all starting at the same place and either show them without duration normalisation, so that the x-axis value is the time in seconds, or with duration normalisation where each sample is another 10% of the way along the x-axis. If we print the dataframe now, we should see this.

```
        Time_s       F0_Hz  Semitones  Cents    ZScore      Time
0   148.392882  213.270192   1.831682    183  0.966044  0.000000
1   148.413310  214.782037   1.953974    195  1.027084  0.020428
2   148.433739  213.038641   1.812876    181  0.956657  0.040857
3   148.454168  210.669344   1.619259    162  0.860016  0.061286
4   148.474597  205.821561   1.216224    122  0.658847  0.081715
5   148.495025  198.473559   0.586856     59  0.344708  0.102143
6   148.515454  187.811133  -0.369115    -37 -0.132450  0.122572
7   148.535883  174.942274  -1.597958   -160 -0.745808  0.143001
8   148.556311  167.902131  -2.309059   -231 -1.100743  0.163429
9   148.576740  162.450427  -2.880511   -288 -1.385974  0.183858
10  148.597169  161.281436  -3.005541   -301 -1.448381  0.204287
```

Here we have our original two columns, `Time_s` and `F0_Hz`, as well as a few others. First, the converted `Semitones` values, where 0 is set to the mean pitch value in Hz. Then `Cents`, which is an integer and corresponds to semitones. The `ZScore` column is based on the semitone values and so will already be logarithmic. Then finally, the `Time` column always starts at 0 for each token, allowing us to use that as the x-axis value for seconds, or use the index of the row multiplied by 0.1 as a percent value. Note the formatting of this printed value will be different from our original file since this is a printed pandas dataframe. This is what we wanted, but there's one more thing we need to do.

### 5.5 Calibrating Values Across Tokens

All of the values are being calculated locally to the token. In this case all of our tokens are of the same word and therefore the same toneme, but what if we wanted to compare different tonemes? A high-to-mid falling tone and a mid-to-low falling tone would be indiscernible the way we've done it. So instead, we need to calculate the z-Score globally rather than locally.

To do this, first we're going to initialise a second dataframe just before our `for` loop:

```
all_data = pd.DataFrame()
```

Also before the `for` loop, we'll add another loop simply to concatenate all of the data files to that new dataframe called `all_data`. We're calculating semitones here, which means doing it twice overall, as a way to ensure we have a global logarithmic scale from which to calculate the z-Score.

```
for filename in filelist:
    data = pandas.read_csv(filename , sep="   ", engine="python")
    data["Semitones"] = 12 * data["F0_Hz"].apply(
        lambda x: math.log(x / data["F0_Hz"].mean(), 2)
    )
    all_data = pandas.concat([all_data, data], ignore_index=True)
```

Then, we can modify the line where we create the `ZScore` column to instead refer to the `all_data` dataframe:

```
    data["ZScore"] = (data["Semitones"]
        - all_data["Semitones"].mean()) / all_data["Semitones"].std()
```

This gets us a slightly different version of the data from our first token in that the z-Scores are now calculated globally, while none of the other values will have changed:

```
        Time_s       F0_Hz  Semitones  Cents    ZScore      Time
0   148.392882  213.270192   1.831682    183  1.444356  0.000000
1   148.413310  214.782037   1.953974    195  1.538367  0.020428
2   148.433739  213.038641   1.812876    181  1.429899  0.040857
3   148.454168  210.669344   1.619259    162  1.281058  0.061286
4   148.474597  205.821561   1.216224    122  0.971229  0.081715
5   148.495025  198.473559   0.586856     59  0.487409  0.102143
6   148.515454  187.811133  -0.369115    -37 -0.247483  0.122572
7   148.535883  174.942274  -1.597958   -160 -1.192143  0.143001
8   148.556311  167.902131  -2.309059   -231 -1.738794  0.163429
9   148.576740  162.450427  -2.880511   -288 -2.178092  0.183858
10  148.597169  161.281436  -3.005541   -301 -2.274207  0.204287
```

Note that the z-Score of the first row has changed from 0.966044 to 1.444356 once we made it calculate the values based on the global rather than local mean. This is what we want, because it means now if we have multiple tonemes, we can see the correct values. However, if we were only looking at a single toneme, it might make sense to keep the z-Score as being based on local values. That's not something we would usually want to do, but there may be reasons to do so.

### 5.6 Wrapping up & Plotting the Data

The final code so far looks like this:

```
import pandas
import math
import matplotlib.pyplot as plt
from pathlib import Path

filelist = Path().glob("./data/*")
all_data = pandas.DataFrame()

for filename in filelist:
    data = pandas.read_csv(filename , sep="   ", engine="python")
    data["Semitones"] = 12 * data["F0_Hz"].apply(
        lambda x: math.log(x / data["F0_Hz"].mean(), 2)
    )
    all_data = pandas.concat([all_data, data], ignore_index=True)

for filename in filelist:
    data = pandas.read_csv(filename , sep="   ", engine="python")
    data["Semitones"] = 12 * data["F0_Hz"].apply(
        lambda x: math.log(x / data["F0_Hz"].mean(), 2)
    )
    data["Cents"] = round(data["Semitones"] * 100).astype(int)
    data["ZScore"] = (
        data["Semitones"] - all_data["Semitones"].mean()) /
all_data["Semitones"].std()
    data["Time"] = data["Time_s"] - data["Time_s"].min()
```

The last thing to do is plot the values. So first, we instantiate the plot before the final for loop.

```
plt.figure(figsize=(10, 6))
```

Then at the end of the final for loop, we need to save the data to the plot.

```
     yscale = "ZScore"
    x = np.linspace(data["Time"].min(), data"Time"].max(), 300)
    spl = make_interp_spline(data["Time"], data[yscale], k=2)
    y = spl(x)

    plt.plot(x, y, label=filename)
```

Finally, after the final for loop and at the end of the script, save the plot.

```
plt.title("log z-Score contours")
plt.xlabel("Time (s)")
plt.ylabel("z-Score")
plt.legend(loc="upper right", fontsize='small')
plt.grid(True)
plt.savefig("./plot.png", dpi=300)
```

This gives us a plot of the pitch contours, normalised for pitch using logarithmic z-Scores. Our plot looks like this for the sample data:

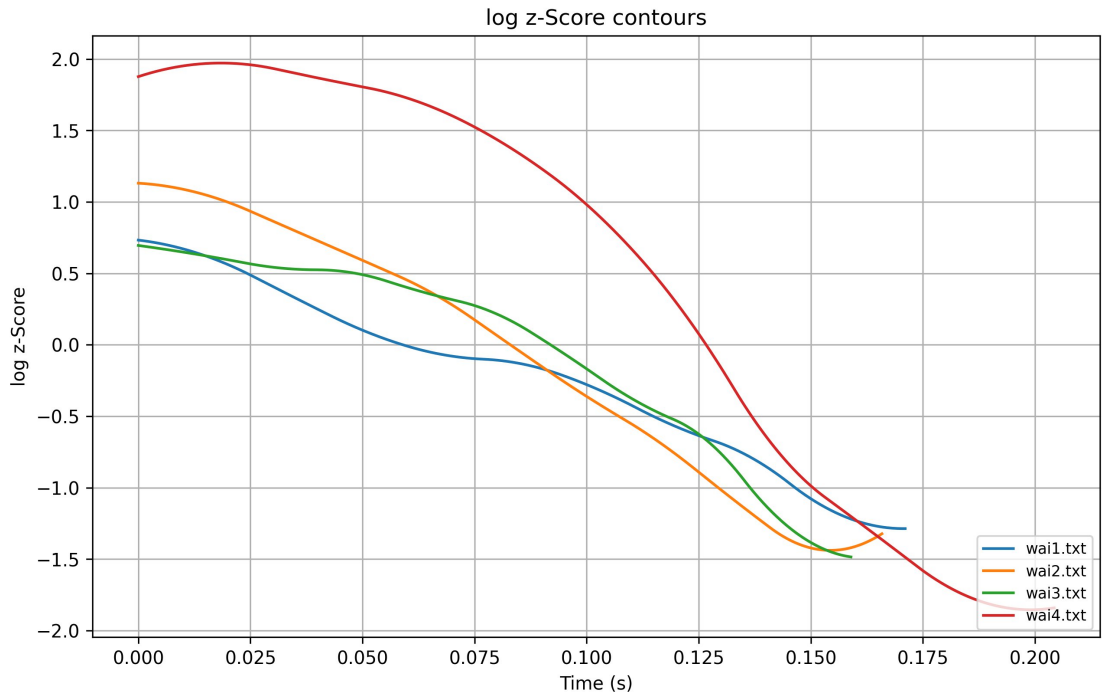


**Figure 1:** Plot of tone contours for *wai* 'bird' in Wolam Khiamniungan

With pandas we could easily also keep track of the original filename and create averages based on lexeme, or any other kind of useful data visualisation technique. And nearly anything we would want to do would only require one or two lines of additional code.

## 6 Conclusion

By using Semitones or logarithmic z-Scores we are able to draw meaningful conclusions from our analysis which would not be available by relying solely on Hz. As human pitch perception is ultimately logarithmic, we must be working in a logarithmic scale when discussing how linguistic pitch occurs. By including Semitone values, we present a salient system familiar to anyone who has even superficial experience with musicality, and by

including logarithmic z-Scores, we ensure cross-speaker normalisation can properly account for differences in pitch space for the entire set of speakers whose speech we may be investigating.

Importantly, whether discussing F0 contours of vowel formant data, work on linguistic pitch should be based on openly available data in multiple formats. Hz values are a must to ensure reproducibility, but are not sufficiently informative when it comes to description or analysis. Ideally, the published data should include multiple scales in the raw data. By including our semitone or log-Z values, we give additional transparency to our methodology for the reader or those wishing to replicate our results. As shown above, the calculation, visualisation, and creation of a multi-scale dataframe from the original Hz values is made easy, thanks to widely used Python packages that require minimal effort or programming knowledge.

## References

Boersma, Paul & Weenink, David (2024). Praat: doing phonetics by computer [Computer program]. Version 6.4.18, retrieved 21 August 2024 from http://www.praat.org/

Chen, Yingshi 陳応時 (1988). “Temperamentology in ancient Chinese written records”. In: Musicology Australia 11.1, pp. 44–64.

van Dam, Kellen Parker (2018). The tone system of Tangsa-Nocte and related Northern Naga varieties (Doctoral dissertation, La Trobe).

Ellis, Alexander John (1885). “On the musical scales of various nations”. In: The journal of the society of arts 33.1688.

Fant, Gunnar (1968). “Analysis and synthesis of speech processes”. In: Manual of phonetics 2, pp. 173–277.

— (2006). Speech acoustics and phonetics: Selected writings. Vol. 24. Springer Science & Business Media.

Glasberg, Brian R, & Brian CJ Moore (1990). “Derivation of auditory filter shapes from notched-noise data”. In: Hearing research 47.1, pp. 103–138.

Greenwood, Donald D (1961). “Critical bandwidth and the frequency coordinates of the basilar membrane”. In: The Journal of the Acoustical Society of America 33.10, pp. 1344–1356.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(03), 90-95.

Jitwiriyanont, Sujinat (2014). “Northern Pa-O Lexical Tones: An Analysis of Polynomial Equations Generated from Semitone Values”. In: Paper presented at RGJ-Ph.D Congress XV Jomtien Palm Beach Hotel and Resort.

Kubik, Gerhard (2010). Theory of African music. Vol. 1. University of Chicago Press.

Makhoul, John, & Lynn Cosell (1976). “LPCW: An LPC vocoder with linear predictive spectral warping”. In: Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’76. Vol. 1. IEEE, pp. 466–469.

McKinney, W. (2010). Data structures for statistical computing in Python. In SciPy (Vol. 445, No. 1, pp. 51-56).

Miller, J. D., Engebretson, A. M., & Vemula, N. R. (1980). Vowel normalization: Differences between vowels spoken by children, women, and men. The Journal of the Acoustical Society of America, 68(S1), S33-S33.

Nolan, Francis (2003). “Intonational equivalence: an experimental evaluation of pitch scales”. In: Proceedings of the 15th International Congress of Phonetic Sciences, Barcelona. Vol. 39.

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. The journal of the acoustical society of america, 8(3), 185-190.

Stevens, Stanley S, & John Volkmann (1940). “The relation of pitch to frequency: A revised scale”. In: The American Journal of Psychology 53.3, pp. 329–353.

Traunmuller, Hartmut (1990). “Analytical expressions for the tono-topic sensory scale”. In: The Journal of the Acoustical Society of America 88.1, pp. 97–100.

Zwicker, Eberhard (1961). “Subdivision of the audible frequency range into critical bands (Frequenzgruppen)”. In: The Journal of the Acoustical Society of America 33.2, pp. 248–248.

| **Supplementary Material** |
| --- |
| Data and code presented in this study have been curated on GitHub (https://github.com/calc-project/blog-pitch, Version 1.0) and archived with Zenodo (DOI: 10.5281/zenodo.13895995). |