Exercises manual for the book *Music Data Analysis: Foundations and Applications*

by

Claus Weihs, Dietmar Jannach, Igor Vatolkin and Günter Rudolph (Eds.)

Version 1 (December 15, 2016)

Department of Statistics and Department of Computer Science
TU Dortmund University, Germany

Acknowledgement

We thank Dr. Klaus Friedrichs for co-editing this manual, and Daniel Neudek, B.Sc. and Jennifer Neuhaus-Stern, B.Sc. for providing solutions.

Chapter 1
Organization of Exercises

In what follows you will find a description of the organization of the exercises.

- Each chapter has simple and more elaborated exercises. Simple exercises can, e.g., be found in Exercises 2.1, 3.1, and 5.1.
- There are theoretical and practical exercises. Practical exercises can, e.g., be found in Exercises 2.2, 9.1, and 11.1.
- The practical exercises either rely on artificial data or data from the **data sets** published on the web site. A list of such data sets and their contents can be found in Appendix A "Databases for Exercises" in this exercise book.
- In order to include the data from the data sets, please
  1. download the file "data_exercisesMDA.zip" from the book website `http://sig-ma.de/music-data-analysis-book` into an internal directory,
  2. unpack the .zip-file, and
  3. replace the dots `...` in the dummy data set name of the code, e.g. in `...\Data\Auditory_Models\`, by the internal directory name.
- The solutions of the exercises are only given in the solution book, which includes theoretical solutions as well as computer programs (in R, Matlab, or Python) and the corresponding results for the practical exercises.
- If you are a university instructor giving a course on "Music Data Analysis" or a related topic, you can ask for the solutions manual by writing an e-mail to `orders@taylorandfrancis.com` indicating your name, position, and the course you are responsible for.
- For the solutions we have prepared (and will continue to do so) R- and Matlab-**functions** to simplify the task. A list of such functions and their code can be found in Appendix B "R- and Matlab-Functions for Exercises" in this exercise book.
- For each chapter there are chapter-specific exercises related to the higher-level topics specified below; such exercises are marked accordingly.
- As higher-level topics we have identified the four musical moments (see Chapter 2 in the book)
  - pitch (including chroma, chords, harmony, and key),
  - volume (including loudness, energy, rests, vibrato, and amplitude modulation),
  - timbre (including MFCC, spectrogram, formants, transients, instrument recognition, emotions), and
  - duration (including rhythm, measure, onset detection, ASDR identification, structure identification, tempo recognition)

  as well as the topic
  - genre (including music recommendation and organization).
- The exercises corresponding to higher-level topics are listed in Appendix C "List of Exercises for Higher-level Topics" in this exercise book.

Contents

Part I

Exercises

Chapter 2

The Musical Signal

(Sebastian Knoche and Martin Ebeling)

Basic Exercises

Exercise 2.1 [Pitch - Equal Temperament]

Equal temperament: What are the frequencies in Hz on the equally tempered scale for $k = -24$ and $k = -20$?

Exercise 2.2 [Timbre - Cylindric Instruments]

The timbre of cylindric musical instruments (with one end closed and one end open) is mainly determined by odd multiples of the fundamental, see Chapter 2.4.7 in the book. Look at the periodograms (magnitude of DFT) of the clarinet and flute sounds in the Spectrum database (see Appendix A.5). Compare the corresponding plots with the theoretical result. How are the results related to the boundary conditions at the two ends of the instruments? For plotting the periodograms in R, you can use the function plot_spectrum() (see Appendix B.11).

Exercise 2.3 [Volume - Sound Pressure Level and Loudness]

One sound source produces, in a given distance, a sound pressure level of $L_{p1} = 80\,\mathrm{dB}$. We add a second source which is identical to the first one. The resulting sound is an incoherent superposition. Which sound pressure level L_{p2} has the combined sound of the two identical sources?

Let us assume that the two sounds are tones around 1000 Hz. Which loudness (on the sone scale) is produced by the first (single) sound source and the two (combined) sources?

Exercise 2.4 [Pitch - Modulation, Psychoacoustics]

Consider an amplitude modulated tone with a carrier frequency of $\omega_c = 1100\,\mathrm{Hz}$, a modulation frequency of $\omega_m = 220\,\mathrm{Hz}$ and a modulation depth of m. Which tonal percepts are presumable?

Advanced Exercises

Exercise 2.5 [Pitch / Volume - Amplitude Modulation and Vibrato]

Amplitude modulation and vibrato are two different aspects of pitch / volume variation. Please consider the following formulas for these aspects:
for vibrato: $\omega(t) = \omega_c + \Delta\omega \cos(\omega_m t)$, where ω_c is the frequency of the pure sine and ω_m is the vibrato frequency (see Formula (2.31) in the book), leading to
$x(t) = A \sin[\omega_c t + \beta \sin(\omega_m t + \phi)], \beta := \Delta\omega/\omega_m$ (see Formula (2.33)), and for
amplitude modulation: $x(t) = A\,[1 + m\,\cos(\omega_m t + \phi)] \sin(\omega_c t)$, where m is the modulation depth and ω_m the modulation frequency, leading to
$x(t) = A\,[\sin(\omega_c t) + \frac{m}{2} \sin([\omega_c - \omega_m]t - \phi) + \frac{m}{2} \sin([\omega_c + \omega_m]t + \phi)]$ (see Section 2.3.4).

Please note that the modulation depth m is independent of the modulation frequency ω_m and that $\omega = f\,2\pi/f_s$, where f is the frequency in Hz and f_s is the sampling frequency.
Please study the periodograms and the plots of the waves for different values for the frequencies ω_c and ω_m and for the other constants, e.g.
$\omega_c = 440$ Hz, $\omega_m = 8$ Hz, $\Delta\omega = 16, \phi = 0, A = 1, m = 1/3,$ and $f_s = 44100$ Hz.
What are the differences between the periodograms of vibrato and amplitude modulation of a tone with frequency ω_c and to the periodogram of the pure sine with frequency ω_c? What can be seen if you plot the pure sine against the vibrato added version and the amplitude modulated version? Use the package tuneR to play the tones.

Exercise 2.6 [Timbre - Cylindric Instruments]

Repeat the calculation of the eigenmodes of a cylindrical instrument, see Chapter 2.4.7 in the book, this time for two open ends, which corresponds to the case of a flute.

At first, show that the function $\phi(z,t) = A\sin(\omega t)\sin(\omega z/c)$ is a solution of the wave equation $\ddot{\phi}(z,t) - c^2\phi''(z,t) = 0$.

As the flute is open at both ends, the pressure is forced to be close to the atmospheric pressure. The appropriate boundary conditions, if the pipe ranges from $z = 0$ to L, thus read $\dot{\phi}(0,t) = \dot{\phi}(L,t) = 0$. Determine the spectrum ω_n of frequencies for which the solution satisfies these boundary conditions. Compare your results with the periodograms you calculated in Exercise 2.2.

Exercise 2.7 [Volume - Wave Equation]

In many cases, sound is radiated by a source in all directions of the three-dimensional space. For such problems, the wave-equation may be written in spherical coordinates, in which a point in space is defined by its radial distance r from the origin, the polar angle θ and the azimuthal angle ϕ. Considering only waves that are independent from the angles θ, ϕ, the three-dimensional wave equation in spherical coordinates reads

$$\frac{\partial^2}{\partial t^2}\phi(r,t) - c^2\frac{1}{r}\frac{\partial^2}{\partial r^2}\left(r\,\phi(r,t)\right) = 0. \tag{2.1}$$

Show that functions of the form $\phi_\pm(r,t) = f(r \pm ct)/r$ are solutions of the wave equation and interpret them. How do the sound pressure $p^* = -\rho_0\partial\phi/\partial t$, velocity $\boldsymbol{v} = \boldsymbol{e}_r\partial\phi/\partial r$ and intensity $\boldsymbol{I} = p^* \cdot \boldsymbol{v}$ scale with the radial distance r?

Exercise 2.8 [Timbre - Musical Acoustics]

Hermann v. Helmholtz was the first to determine the spectrum of a bowed string. He attached a tiny grain of cornstarch to a string and observed its movements through an *oscilloscope* (a special microscope he had constructed) when the string was bowed. He saw certain Lissajous figures from which he concluded that the motion of a bowed string equals a sawtooth wave. He could read the spectrum of the bowed string from the Fourier series of the sawtooth wave. Try and do the same.

Chapter 3

Musical Structures
(Martin Ebeling)

Basic Exercises

Exercise 3.1 [Pitch - Interval]

Which musical interval corresponds to the two frequencies in Exercise 2.1? Does this interval represent a consonance or a dissonance?

Exercise 3.2 [Pitch - Intervals, Consonance]

The Generalized Coincidence Function (GCF) shows the different degrees of neuronal coincidence of the musically relevant intervals within the range of an octave. Order these intervals according to their degrees of neuronal coincidence as can be read from the GCF (from the highest to the lowest degree) and add the vibration ratios of the intervals. Compare this with the intervals between the harmonics of the overtone series.

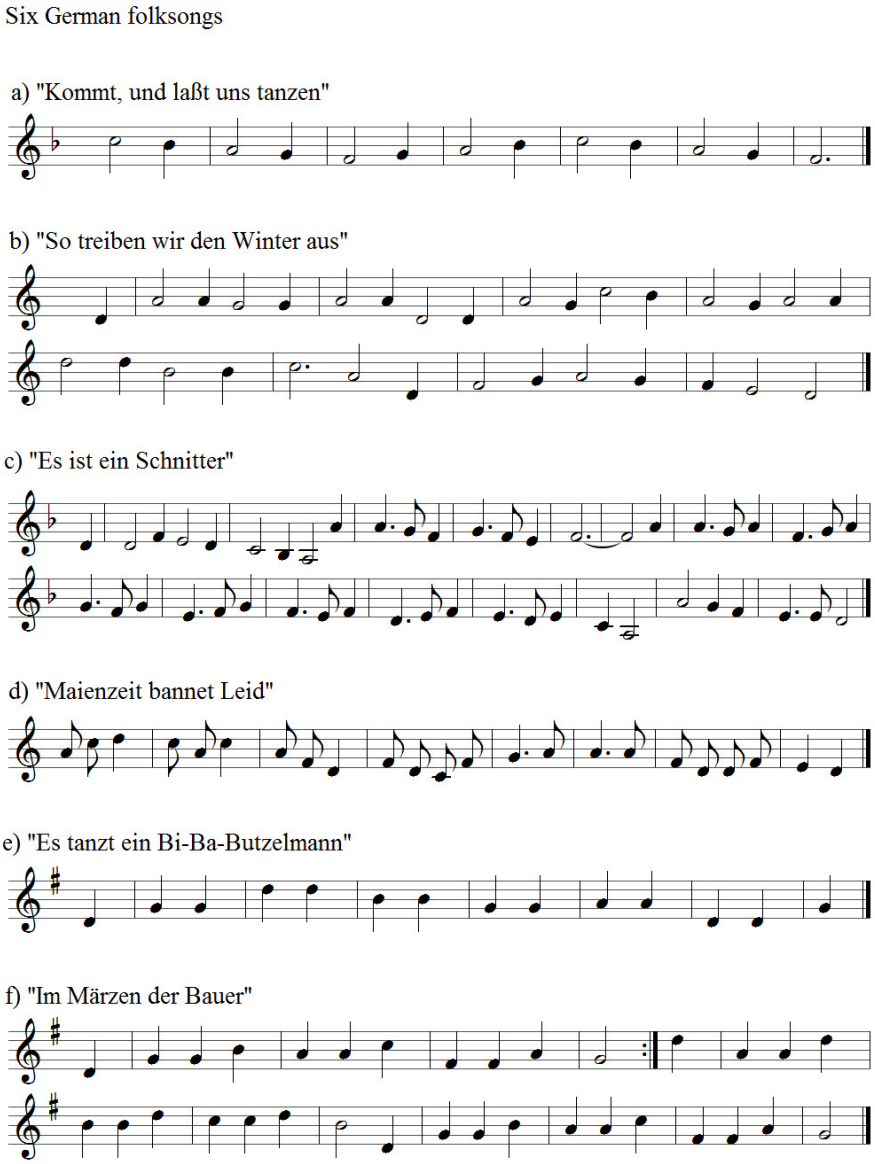

Figure 3.1: Six German folksongs

Exercise 3.3 [Duration / Pitch - Measure, Meter, Key of Melodies]

Fig. 3.1 shows six German folksongs. Determine their measures, meters, and keys.

Exercise 3.4 [Pitch - Tuning Systems]

Over centuries, the issue of tuning systems had extensively been debated until in the 19th century the equal temperament became a commonly accepted compromise.

a) Compute the differences in cent between the pure intonation and the equal temperament of the intervals of Exercise 3.2. Which differences are audible?

b) In the Pythagorean system the twelve chromatic tones are deduced from the pure fifth. Starting from the tone c, twelve successive fifths end up on the tone *his* which corresponds to the tone c under an enharmonic change. Compute their differences in cent.

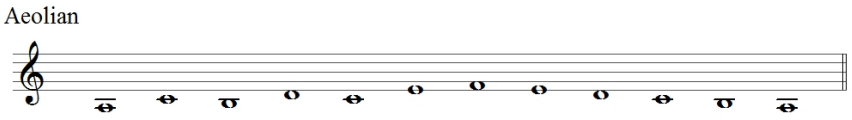

Figure 3.2: The aeolian modus of fux

.

Advanced Exercises

Exercise 3.5 [Pitch - Counterpoint]

To compose a good vocal line against a *cantus firmus* following the rigid rules of the craft of counterpoint is not an easy task. Fig. 3.2 shows Fux's aeolian mode. Write 1:1 counterpoints with two parts. The *cantus firmus* can be the melody of the upper or lower part.

Exercise 3.6 [Pitch/Volume/Timbre/Duration - Gestalt]

To conceive a *Gestalt*, its structure must become obvious already at the moment of perception. The comprehension of all parts of a *Gestalt* and their mutual relations are natural and self-evident. The famous German Philosopher and co-founder of the scientific psychology Carl Stumpf (1848-1936) stated that there are four basic relations:

- plurality - *Mehrheit*
- likeness - *Ähnlichkeit*
- increase - *Steigerung*
- fusion - *Verschmelzung*

Discuss these relations in the context of the four tonal moments.

Exercise 3.7 [Pitch - Harmony]

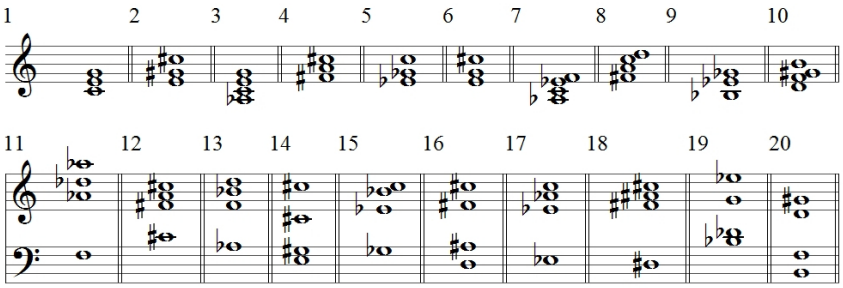

Figure 3.3: Chords in root position or inverted.

Analyse the chords of figure 3.3: determine the root, the type of chord, and its inversion. Further consider which keys these chords could belong to.

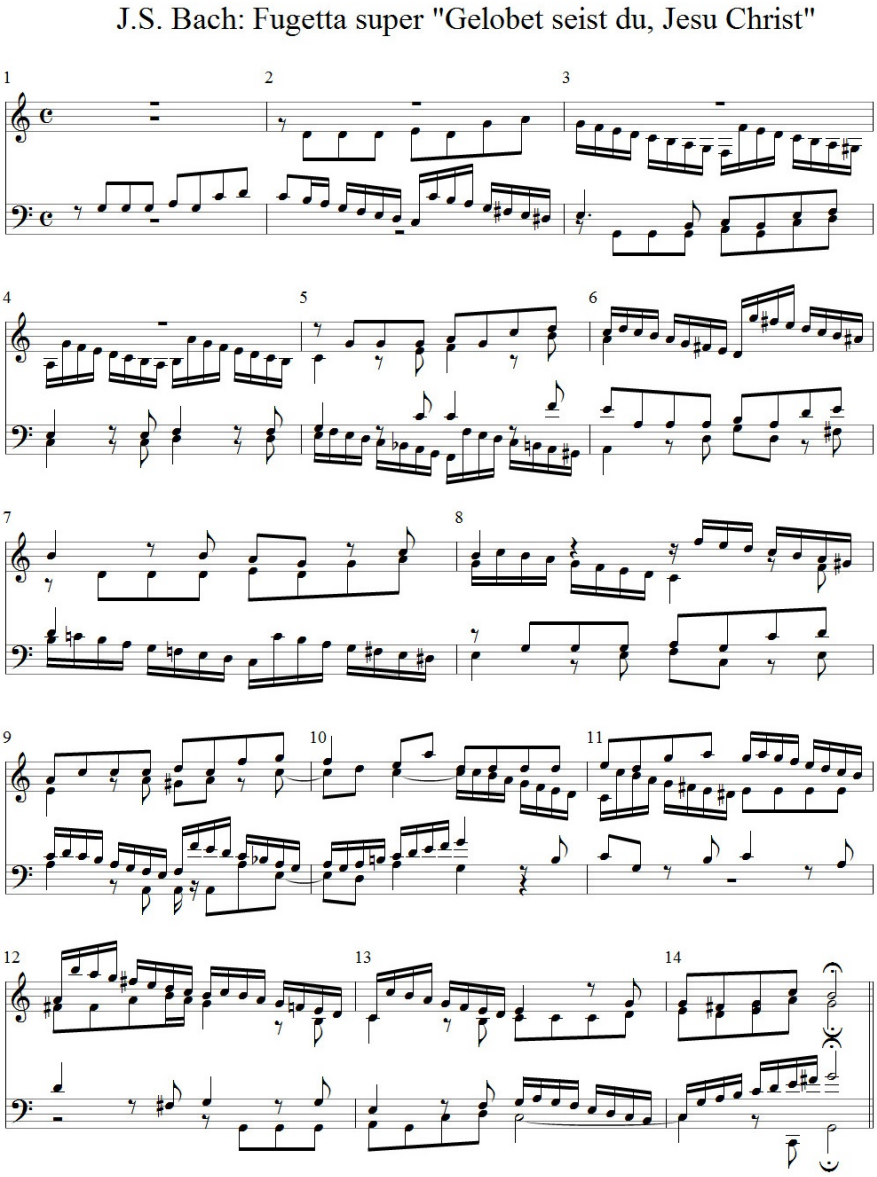

Figure 3.4: J.S. Bach: *Fugetta super "Gelobet seist Du, Jesu Christ"*

Exercise 3.8 [Pitch/Volume/Timbre/Duration - Analysis]

The *Fugetta super "Gelobet seist Du, Jesu Christ"* by J.S. Bach (see Fig. 3.4) is a choral prelude which introduces the choral sung by the congregation during the religious service. It is a masterpiece of baroque counterpoint. The first notes of the choral serve as a theme of a little fugue to which Bach adds a lively counterpoint in another voice. Find out all entries of the theme and the counterpoint. Which is the key of the prelude? On which harmony does it finish and why?

Chapter 4

Digital Filters and Spectral Analysis

(Rainer Martin and Anil Nagathil)

Audio data used in these exercises:
Exercises 4.4, 4.6 and 4.7 use clip 20 of the "1000 Songs Database" (Soleymani, M., Caro, M., Schmidt, E. , Sha, C. and Yang, Y., CrowdMM workshop, ACM Multimedia 2013.) Alternatively, you can also use your own audio signals. For exercise 4.4 the sampling frequency is $f_s = 16$ kHz while for the other two exercises the sampling frequency has to be $f_s = 32$ kHz.

Basic Exercises

Exercise 4.1 [Pitch - Linear System]

A first-order linear time-invariant system is specified via the difference equation

$$y[k] = x[k] - x[k-1]\,. \tag{4.1}$$

- Compute the impulse response and the frequency response as a function of normalized frequency $\Omega = \frac{2\pi f}{f_s}$.
- Plot the magnitude response and the phase response in the range $-2\pi \leq \Omega \leq 2\pi$.
- Is the phase response of this system linear? Compute the group delay.
- Discuss possible applications of this system.

Exercise 4.2 [Pitch / Timbre - Filter Design]

A music signal is sampled at a sampling frequency of $f_s = 16$ kHz. We like to design a low-pass filter and a high-pass filter to extract the lower third and the upper third of the available audio bandwidth, respectively. The filters shall have a minimum stopband attenuation of 80 dB. The width of the transition band is 1 kHz.

- How large is the maximum audio bandwidth of the music signal?
- Use the modified Fourier approximation and the Kaiser window to design the filters. In a first step, estimate the required filter order. Plot the impulse response of the filters and the frequency responses, the latter separately for the magnitude and the phase.
- Use the Chebychev approximation to design the filters. In a first step, estimate the required filter order. Plot the impulse responses of the filters and the frequency responses, the latter separately for the magnitude and the phase.
- Do the resulting filters have the linear-phase property? Compute the group delay of the filters.

Hints:

- In MATLAB, you could use the `signal processing toolbox`, which includes functions like `fir1()` and `firpm()` to design the filters. The window function is generated using the function `kaiser()`. Plots of the frequency response are obtained via the function `freqz()`. Alternatively, you could use the `fdatool` which provides a GUI to access the filter design tools in a user-friendly way.

Exercise 4.3 [Pitch - Spectral Analysis]

A musical note has a fundamental frequency of 220 Hz and is processed with a sampling frequency of $f_s = 32$ kHz. Specify the minimum length of the DFT such that the fundamental frequency and its harmonics can be resolved, assuming that either a rectangular window or a Hann window is employed in the spectral analysis.

Exercise 4.4 [Timbre - Signal Power]

Apply the filters of Exercise 4.2 to a music signal and compute running estimates of the short-term signal power in the low frequency band and in the high frequency band. These power estimates should be implemented in terms of first-order recursive systems according to

$$P[k] = \alpha P[k-1] + (1-\alpha)x^2[k] \tag{4.2}$$

where $x[k]$ denotes the lowpass-filtered or highpass-filtered signal and $\alpha \approx 0.9$ is the smoothing parameter.

- Plot the waveform of the music signal and the short-time power of the two frequency bands as a function of time.
- Relate the estimated signal powers to the music signal and interpret your results.

Advanced Exercises

Exercise 4.5 [Timbre - Spectral Transforms]

Produce the **DFT** and the **Constant-Q-transform** (CQT) spectra of pure sine tones with the fundamental frequencies $f = 2^{-2}440$ Hz and $f = 2^{-5/3}440$ Hz, and of the superposition of these tones. Discuss the differences. The sampling rate and the DFT length should be set to $f_s = 44.1$ kHz and $M = 1024$, respectively. For the CQT, the smallest analysis frequency is set to $f_{\min} = 55$ Hz and the largest analysis frequency to the Nyquist rate. The resolution of the CQT is set to one CQT band per semitone.

Exercise 4.6 [Timbre - Filter Bank]

In this Exercise we will decompose a music signal into frequency subbands using a bank of 32 digital filters. The music signal is sampled at a sampling rate of $f_s = 32$ kHz.

- Using the methodology outlined in Section 4.6.1, design a filter bank with 32 filters which are uniformly spaced across the full frequency range up to the sampling frequency. These filters shall add up to an overall response with unit gain and achieve a sideband attenuation of at least 60 dB. You may use a Kaiser window with an appropriate shape parameter in this design. The impulse response of the prototyp lowpass filter shall have 101 coefficients.
- Plot the individual and overall frequency responses.
- Plot a spectrogram of a music signal using this filter bank.

Exercise 4.7 [Timbre - Gammatone Filter Bank]

In this exercise we like to design a bank of 4th-order Gammatone filters as outlined in Section 4.6.2. The filter bank should be comprised of 21 filters. The center frequencies of these filters should be arranged at intervals one ERB (*equivalent rectangular bandwidth*) with the lowest band at 10 ERB and the highest band at 30 ERB. The mapping of ERB scale to frequency shall be given by

$$f = \frac{10^{\frac{\text{ERB}}{21.4}} - 1}{0.00437}\,. \tag{4.3}$$

- Compute a vector of center frequencies (in Hz) for the filter bank as specified above.
- Compute the impulse responses of the 21 Gammatone filters.
- Plot the individual frequency responses for all filters and overall frequency response.

- Plot a spectrogram of a music signal using this filter bank and compare your result to exercise 4.6.

Hints:

-

Exercise 4.8 [Pitch - Autocorrelation]

Show, how fundamental frequencies can be identified by means of the short-time **autocorrelation function** (ACF) (Equation (4.43)) and the **mean squared difference** (YIN algorithm) (Equation (4.44)), when decaying sinusoidal signals with frequencies $f = 2^{-2}440$ Hz and $f = 2^{-5/3}440$ Hz are given. The sampling frequency of these signals is set to $f_s = 44.1$ kHz. The amplitudes of these signals decay according to $(0.95)^{\frac{kf}{f_s}}$.

First, generate and plot these signals. Then, apply ACF and YIN individually to the two sinusoidal signals and to their superposition. Try $N = 500, N = 1000$ and $N = 2000$ for the number of signal samples. Apply ACF and YIN also to the superposition of $f = 110$ Hz and its 3^{rd} and 5^{th} (odd) partials. Is there a difference to the results above?

Hints:

- The R package `tuneR` includes some helpful functions to create sinusoidal tones.
- Additionally, we have prepared R functions `ACF` and `YIN` for the autocorrelation function and the mean squared difference.
- In MATLAB, you could use the `signal processing toolbox`, which includes functions like `xcorr()` and `findpeaks()`.

Chapter 5

Signal-level Features
(Anil Nagathil and Rainer Martin)

Basic Exercises

Exercise 5.1 [Pitch - Mel Scale]

What are the mel frequencies of the tones with the fundamental frequencies in Exercise 2.1?

Exercise 5.2 [Timbre - Zero Crossings]

Consider a sinusoidal signal and a noise signal. Explain why the noise signal has a higher zero-crossing rate.

Exercise 5.3 [Timbre - Clarinet]

What value do you expect for the even-harmonic energy ratio when you compute it for clarinet sounds?

Exercise 5.4 [Duration - Onset Detection]

Explain why the spectral flux can be used to detect note onsets and for what type of instruments it is particularly suited.

Advanced Exercises

Exercise 5.5 [Timbre - Spectral Centroid]

Assume, that the discrete magnitude spectrum of a harmonic tone can be modeled by

$$A[\mu] = \sum_{n=1}^{N} a^{n-1}\, \delta[\mu - n\, K_0],$$

where N denotes the number of harmonics to be considered, a determines the attenuation of the harmonics with $0 < a \leq 1$, K_0 denotes the frequency bin corresponding to the fundamental frequency of the tone, and

$$\delta[\mu] = \begin{cases} 1, & \mu = 0, \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

is the Kronecker delta.
a) Sketch $A[\mu]$ for
(1) $K_0 = 20$, $a = 0.9$, $N = 10$,
(2) $K_0 = 30$, $a = 0.5$, $N = 5$, respectively.
b) Explain why $A[\mu]$ is not a valid harmonic model for a clarinet tone. How would you modify the model to make it suitable for clarinet sounds?
c) Derive a closed-form expression of the spectral centroid as a function of K_0, N, and a, with $a < 1$.
d) Compute the values of the spectral centroid for the sketched examples.

Exercise 5.6 [Timbre - Spectral Spread and Skewness]

Write a function in R or MATLAB which first computes the STFT representation of an input signal and then calculates the spectral spread and the spectral skewness. Choose $f_s = 44100$ Hz, $M = 2048$, and $R = 64$ as the sampling frequency, the segment length, and the segment shift, respectively.

Use this function to obtain the spectral spread and skewness values for clarinet sounds, oboe sounds, and violin sounds with different fundamental frequencies. Visualize these features using a scatterplot and discuss whether they can distinguish sounds from the three instruments.

Exercise 5.7 [Timbre - MFCCs]

Explain which property of a signal is described by MFCCs and why it makes sense to use MFCCs to characterize timbre.

Exercise 5.8 [Pitch - Chroma]

a) Write a function in R or MATLAB which computes the Chroma and CENS features of a music signal based on its STFT representation. Choose $f_s = 44100$ Hz, $M = 2048$, and $R = 64$ as the sampling frequency, the segment length, and the segment shift, respectively. For computing the CENS features use a decimation factor of 20.
b) What is the resulting feature rate of the Chroma and CENS features, respectively?
c) Use your function to obtain the Chroma and CENS feature series for isolated notes played on different instruments. Explain why often two notes are prominent in the chromagram although only one note is played.

Chapter 6

Auditory Models
(Klaus Friedrichs and Claus Weihs)

Basic Exercises

Exercise 6.1 [Pitch - Sine Tone]

Please download the auditory model of Meddis (MATLAB code)[1]. In the standard settings of the model three different fiber types with different thresholds are considered. For simplification, we are only interested in one of these types. To change this, open the file *userPrograms/runMAP1_14.m* and make the following modification: In line 93 set

```
paramChanges={'IHCpreSynapseParams.tauCa= 86e-6;'};
```

Generate a model output for a pure sine tone with the following properties (parameters are named as in the code): $F0 = 440Hz$, $duration = 0.1s$, $sampleRate = 44100Hz$, $rampDuration = 0.005s$, $leveldBSPL = 50$, $beginSilence = 0.01s$ and $endSilence = 0.1s$.

Exercise 6.2 [Pitch - Harmonic Tone]

Generate a model output for a harmonic tone (6 partials with identical intensities) with the following properties: $F0 = 440Hz$, $duration = 0.1s$, $sampleRate = 44100Hz$, $rampDuration = 0.005s$, $leveldBSPL = 50$, $beginSilence = 0.01s$ and $endSilence = 0.1s$. What are the main differences of the output compared to the sine tone of Exercise 6.1?

Exercise 6.3 [Pitch - Sine Tone]

Consider the release rates (neural activity) of the auditory model for a pure sine tone with a frequency of 440 Hz. Visualize the mean release rates of each channel. Which one has the hightest activity? Can you explain the result?

Hint: Type in the command window

```
global   ANprobRateOutput
```

before you run the model in order to get the matrix of the release rates.

Exercise 6.4 [Pitch - DFT]

Consider again a pure sine tone with a frequency of 440 Hz. Visualize the DFT magnitude for the release rates of the channels 1, 9, 15, 26 and 41. Are there any differences between the channels outputs? Please discuss the result.

[1] http://www.essexpsychology.macmate.me/resources/software/MAP1_14h-public.zip

Chapter 6 – Advanced Exercises

Exercise 6.5 [Pitch - Harmonic Tone]

Consider the synthetic complex tone of Exercise 6.2. Repeat the tasks of Exercise 6.3 and Exercise 6.4. What are the differences in comparison to the pure tone? Repeat the Exercise for a tone with 5 overtones.

Exercise 6.6 [Timbre - Real Tones]

Consider real tones of cello, clarinet, flute and trumpet (A4 = 440 Hz). You can find the release rates for such tones in the files "cello_ANoutput.csv", "clarinet_ANoutput.csv", "flute_ANoutput".csv and "trumpet_ANoutput.csv". Repeat the tasks of Exercise 6.3. What are the differences between the different instruments? Please compare the results also to the synthetic tones of Exercise 6.5.

Exercise 6.7 [Pitch - Fundamental Frequency]

Show, how the fundamental frequency can be identified by means of the summarized autocorrelation function (**SACF**) as given in Equation 6.2. Exemplify this for the harmonic tone of Exercise 6.2. Visualize the SACF for lags $l = 0, 1, \ldots, 1000$. Which frequency corresponds to the maximum peak?

Hint: In MATLAB the function `xcorr(y, maxLag)` of the `Signal Processing Toolbox` returns the autocorrelation sequence for the range from -maxlag to maxleg.

Exercise 6.8 [Timbre - Characteristics]

Most of the features (characteristics) discussed in chapter 5 can be adopted to the auditory output in a channel-wise manner. Consider the feature spectral centroid (see Definition 5.3 in chapter 5). Extract this feature for the complex tone of Exercise 6.2 from all 41 channels seperately. Discuss the differences of the feature for the different channels. How do these values change if the tone consists of 1 or 3 partials? How does they change to a transposition to 220 Hz?

Hint: You can use the MIRtoolbox[2] for feature extraction.

[2] `https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox`

Chapter 7

Digital Representation of Music
(Günter Rudolph)

Basic Exercises

Exercise 7.1 [Pitch - Helmholtz System]

What are the labels of the tones with the fundamental frequencies in Exercise 2.1 in the Helmholtz system? What are the MIDI numbers for the fundamental frequencies in Exercise 2.1?

Exercise 7.2 [Pitch - Key]

Suppose you want to specify the key A major as a key signature event in the header of a standard MIDI file. How many bytes are necessary and which are their values?

Exercise 7.3 [MP3 Format]

Suppose you have detected a candidate for a frame header of an MP3 file or data stream with binary representation

```
1111 1111 1111 1011 0111 0100 1100 1100.
```

Argue why these 32 bits cannot be rejected as a possible MP3 frame header.

Exercise 7.4 [Duration - MIDI]

Explain the concept of note length in MIDI and how to specify half, quarter, and eighth notes assuming nonuse of the LTC time code.

Advanced Exercises

Exercise 7.5 [Pitch - Key]

Transpose the melody of Example 7.1 in the book to G major and change the file in abc format accordingly. Use some tool like `abcm2ps` to view the score.

Exercise 7.6 [Duration - Quantization]

Which sequence of octal code words results from the quantization of Example 7.4 in the book if we use 8 instead of 16 levels and a doubled sample interval?

Exercise 7.7 [Data Compression]

Suppose you have detected a frame header of an MP3 data stream with binary representation

```
1111 1111 1111 1011 0111 0100 1100 1100.
```

If the compressed LPCM-encoded data stream has 2^{16} quantization levels, which compression ratio has been realized?

Exercise 7.8 [Duration - Meter]

Change the MusicTEX code of Example 7.6 in the book, such that the duration of each note and pause is halved. Moreover, change the meter to 2/4. Show the score generated from your code.

Chapter 8

Music Data: Beyond the Signal Level

(Dietmar Jannach, Igor Vatolkin, and Geoffray Bonnin)

Basic Exercises

Exercise 8.1 [Semantic Features]

What is meant by "semantic features" (in contrast to signal-level features)? List examples of such semantic features. Give examples in which such semantic features are helpful in music applications.

Exercise 8.2 [Deriving High-Level Features]

Describe in general terms how supervised classification techniques and in particular the Sliding Feature Detection technique can be used to derive high-level features.

Exercise 8.3 [Types of Additional Information]

Analyze the public APIs of the developer websites of last.fm (http://www.last.fm/de/api) and The Echo Nest (http://developer.echonest.com/). Describe which types of information can be accessed from these sites and discuss commonalities and differences.

Exercise 8.4 [Relevance of Terms in Lyrics]

Describe the general principle of the TF-IDF representation of text documents.

Advanced Exercises

Exercise 8.5 [Tags]

The file *playlists50.csv* contains 50 playlists with Last.FM listener tags and EchoNest features. Write the MATLAB code which estimates the most frequent 15 tags. Can these tags be used for genre or emotion prediction? For music recommendation? Discuss the problems.

Exercise 8.6 [Tags, Regression]

1. *playlists50.csv* stores values for several EchoNest features and strengths of tags. Write the MATLAB function which searches for the tag strengths and values of EchoNest features in *playlists50.csv*. Use `function val = findValue(string,name)` as the header (in the *string* it is searched for the value of a feature or the strength of a tag with a given *name*). When the tag is not found, set the output to NaN (not a number).

2. Extend the code from Exercise 8.5 to measure the regression between 7 EchoNest features and 15 most frequent tags. Set the feature names with the following code:

```
featureNames = {'"hotttnesss"','"loudness"','"tempo"','"danceability"', ...
'"key"','"mode"','"energy"'};
```

Use only observations for regression for which the tag strengths are existing. Output the three combinations of a tag and a feature with the highest R^2 value (goodness of fit, cf. Definition **??**). For the regression, use the function `[~,g] = fit(currFeature,currTag,'poly1')`.

Exercise 8.7 [Lyrics]

Load the file *lyrics50.csv* with lyrics for 50 playlists. Write the MATLAB code for the estimation of TF-IDF for the following words: always, but, christmas, city, feel, love, think, you, world, yeah. For the description of TF-IDF, see Section 8.7. Instead of normalization by the frequency of other words after Equation 8.1, normalize by the overall number of occurrences of the term across all lyrics.

Exercise 8.8 [Genre - Correlation to Characteristics]

In the final exercise, let us calculate the correlation between different music descriptors. Write the MATLAB code, which measures the correlation between all possible pairs between words represented by their TF-IDF values, features, and tags.

- Estimate TF-IDF for ten words from the previous exercise.
- Load the following EchoNest features as in Exercise 8.6:

```
featureNames = {'"hotttnesss"','"loudness"','"tempo"','"danceability"', ...
    '"key"','"mode"','"energy"'};
```

- Load the most frequent 10 tags:

```
tagNames = {'"rock"','"alternative"','"favorites"','"pop"','"alternative rock"', ...
    '"indie"','"Love"','"Awesome"','"male vocalists"','"90s"'};
```

Output three most correlated tag and feature, three most correlated tag and TF-IDF value, three most correlated feature and TF-IDF value.

Chapter 9

Statistical Methods
(Claus Weihs)

Basic Exercises

Exercise 9.1 [Genre - Blues]

For the Jazz Blues version given in Example 9.3, give the probabilities of chords related to the tonic I, the subdominant IV, and the dominant V, basic for blues schemes. As a unit use half a bar. Compare the distributions of the chords related to I, II, III, IV, V, and VI with the corresponding distribution of the Standard blues scheme by means of a bar chart as in Figure 9.1. What is the modal value of the two distributions of chords? If we think of the chords as ordered by their number, what are the 50%-quantiles $q_{0.5}$ of the two distributions? Also give the probability for a 7th chord. Interpret the results.

Exercise 9.2 [Genre - Blues]

Are the distributions in Exercise 9.1 uniform distributions? Please explain! Give a variation of the standard blues scheme, where the involved chords are uniformly distributed.

Exercise 9.3 [Pitch - Periodogram]

Show, how the fundamental frequencies in Exercise 2.1 can be identified analogue to Exercise 4.8 by means of the **periodogram** instead of the original time series. Calculate the periodogram corresponding to the original time series of the two individual frequencies and of their superposition.

Hint: `periodogram()` in library `tuneR` calculates the periodogram of a time series.

Exercise 9.4 [Timbre - MFCC]

Show that a pure sine can be easily distinguished from a sine with overtones by means of the first two MFCCs. Use, e.g., $f = 110$ Hz as the frequency of the pure sine, and the 3^{rd} and 5^{th} (odd) partials as overtones. Try MFCC 1 only as well as MFCC 1 vs. MFCC 2. Use a sampling rate of 44100 Hz.

Advanced Exercises

Exercise 9.5 [Timbre - Instrument]

Discuss whether **guitar tones** and **piano tones** can be completely distinguished by means of the first MFCCs. Use the database `svm.RData` (see Appendix A.1) with MFCCs from 4309 guitar tones and 1345 piano tones (see book, Example 9.9). Use MFCCs from block 1 and block 5 of the tones, i.e. from the beginnings and the ends. Beginnings and ends of tones are best suited for instrument recognition because of different onset and offset sounds.
Plot MFCC1 in block 1 (start of tone) and block 5 (end of tone) for the guitar tones 1 - 50 and the piano tones 1-50 (with the row-indices 271-320 in the data matrix). Compare both, the blocks and the instruments. Also compare the variation of MFCC for the different tones. Can guitar and piano be distinguished by means of this information? What could be a measure for distinction?

Exercise 9.6 [Timbre - Instrument]

As in Exercise 9.5, discuss whether **guitar tones** and **piano tones** can be completely distinguished by means of the first MFCCs. Use, again, the database `svm.RData` with MFCCs from 4309 guitar tones and 1345 piano tones (see book, Example 9.9) and the MFCCs from block 1 of the tones, i.e. from the beginnings. Beginnings of tones are best suited for instrument recognition because of different onset sounds. This time, plot MFCC 1 vs. MFCC 2. Does this help for the distinction of guitar and piano?

Exercise 9.7 [Timbre - Instrument]

Reconsider Exercise 9.5 and test whether MFCC 1 in block 1 and block 5 have the same expected value for guitar as well as piano. Use the same data as in Exercise 9.5. Discuss the results, in particular the p-values.

Exercise 9.8 [Timbre - Instrument]

Reconsider Exercise 9.5 and carry out a regression analysis of MFCC 1 in block 5 on block 1 for the selected observations of guitar and piano. Compare the results with the results of the above Exercise.

Chapter 10

Optimization
(Günter Rudolph)

Basic Exercises

Exercise 10.1 [Multi-objective Feature Selection]

Suppose we want to find a feature set for some fictitious classifier endowed with maximum accuracy and the least number of features. Application of some multi-objective algorithm led to a solution set graphically represented in Figure 10.1. Which of the solutions with labels A, B, ..., Q are not dominated?

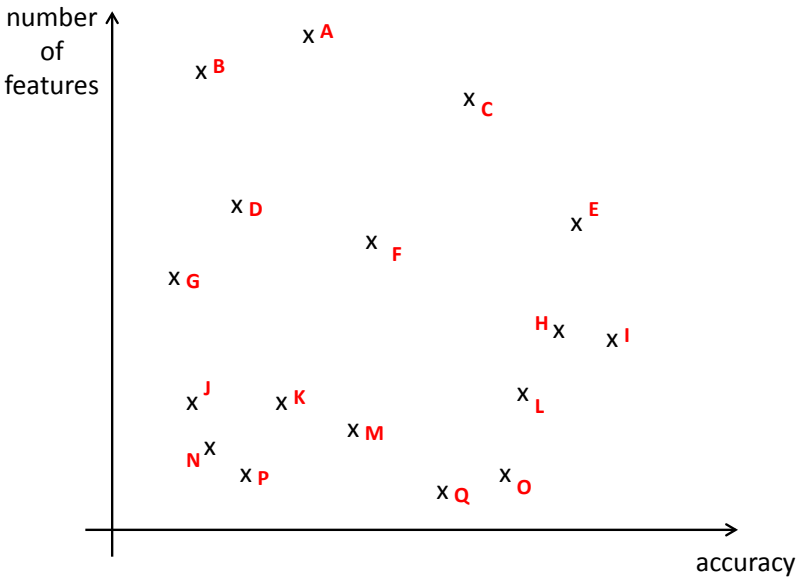

Figure 10.1: Fictitious solution set in objective space.

Advanced Exercises

Exercise 10.2 [Single-objective Feature Selection]

Suppose we want to find a set of features that maximizes the precision of some classifier. The features are labeled with indices from 1 to n. If a feature with index i is use for classification we set variable $x_i = 1$, otherwise $x_i = 0$. To save time and effort we assume that the objective function

$$f(x) = \frac{15 - |s(x) - 15|}{20} \cdot \left| \sin\left(\frac{\pi}{2} \cdot s(x)\right) \right| \quad \text{with} \quad s(x) = \sum_{i=1}^{20} x_i$$

provides the precision $f(x) \in [0, 1]$ for any $x \in \{0, 1\}^{20}$. In reality, we do not know the objective function explicitly and its evaluation requires the time-consuming training of a classifier and its evaluation.

Apply the NEXT and BEST IMPROVEMENT heuristics with starting point $(1, 0, 1, 0, 1, 0, \ldots, 1, 0\}$. Which are the best solutions found by these heuristics? Can we do better?

Chapter 11

Unsupervised Learning
(Claus Weihs)

Basic Exercises

Exercise 11.1 [Pitch - Distance, Blues]

Calculate distances of the chords in the Standard Blues scheme I, I, I, I, IV, IV, I, I, V, V, I, I and in the Jazz Blues version I7, IV7 IVdim, I7, Vm7 I7, IV7, IVdim, I7, III7 VI7, IIm7, V7, III7 VI7, II7 V7 given in Example 9.3 (cf. Exercise 9.1). Define **chord distance** as the distance on the 1D line of fifths c, g, d, a, e, h, f sharp, c sharp, a flat, e flat, b flat, f (cf. Section 3.2.2). Note that this distance is only based on the roots of the chords, ignoring important properties like major or minor and sevenths. Use full bars as the unit for the Standard blues scheme and half bars as the unit for the Jazz blues scheme. Assume I = C.

Exercise 11.2 [Pitch - Distance, Blues]

Calculate distances between the Standard Blues scheme I, I, I, I, IV, IV, I, I, V, V, I, I and the Jazz Blues version I7, IV7 IVdim, I7, Vm7 I7, IV7, IVdim, I7, III7 VI7, IIm7, V7, III7 VI7, II7 V7 given in Example 9.3 (cf. Exercise 9.1). Use generalized versions of the Hamming distance and the simple matching index, where the entries in the observations are allowed to be generally nominal, i.e. can assume more than two values. Use half bars as the unit and base distance only on chord roots. Assume I = C.

Exercise 11.3 [Pitch - Chord Distance]

The distance measures in Exercise 11.1 and Exercise 11.2 are too simple in two respects, Exercise 11.1 because only chord roots are used, Exercise 11.2 because musical properties are fully ignored. In this Exercise we will cure these drawbacks by introducing a much more realistic (but also more complicated) Chord distance rule proposed by [3].

First define the distance $d(x, y) = j + k$ between two chords x and y, where j = minimal number of applications of the Circle-of-fifths rule needed to shift the root level of x into y, and k = sum of the number of non-common pitch classes in the levels within the basic spaces of x and y divided by 2. A pitch class is non-common if it is present in x or y but not in both chords. The basic space of a chord is the scale of tones $\{0, 2, 4, 5, 7, 9, 11\}$ in $Major$ keys and $\{0, 2, 3, 5, 7, 8, 10\}$ in *minor* keys, where 0 corresponds to the root of the key. One application of the Circle-of-fifths rule corresponds to a shift of the root level of the chord x 4 steps to the right or to the left on the basic scale of x (modulo 7). Note that $0 \leq j \leq 3$. If a root level of y is not on the basic scale of x, then set $j = 3$.

As an example calculate the distance between $x = F = \{F, A, C\}$ and $y = G^7 = \{G, B, D, F\}$.

Exercise 11.4 [Duration - Clustering]

Let us now cluster the following 1D-data X of lengths of music pieces (in min) by means of the hierarchical single linkage method: $X = \{0.9, 1, 1.15, 1.8, 2.0, 2.4, 2.7, 3.05, 3.6\}$. Apply the single linkage method to the prior 9 individual clusters. Report all steps of the method. How many clusters would you prefer? Where do you want to split the clusters?

Advanced Exercises

Exercise 11.5 [Pitch - Chord Distance]

Let us consider the basic chord progressions of five songs in $FMajor$ from The Real Vocal Book (Volume II, High Voice), namely

I. Ac-cent-tschu-ate the positive,

II. Aren't you glad your're you,

III. As long as I live,

IV. At sundown, and

V. Be careful, it's my heart.

The corresponding basic 8-bar chord progressions are given in Table 11.1:

Table 11.1: Chord Progressions of the songs I - V; 1 beat = 1 quarter, $F^{j7} = F^{maj7}$

beat	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
I.	F	F^{+}	F^{6}	F	G^{-7}		C^{7}		F	F^{+}	F^{6}	F	G^{7}	C^{7}	F^{6}	
II.	F	F/A	B^{b6}	B^{07}	C	C^{7}	F^{6}	D^{-7}	G^{-7}	C^{7}	F^{6}	D^{7}	G^{7}	C^{7}	F^{6}	
III.	F^{j7}		A^{7}		D^{7}				G^{7}		C^{7}		F^{6}		F^{6}	
IV.	G^{-7}		C^{7}		F^{6}	B^{b7}	A^{-7}	D^{7}	G^{-7}		C^{7}		F^{6}		F^{6}	
V.	F^{07}		F^{j7}		F^{07}		F^{j7}		E^{7}/F		F^{j7}		G^{7b9}	C^{7}	F^{6}	

For these progressions, identify the involved chords, the corresponding basic scales, and the corresponding distances to the triadic tonic chord x of the key of all songs $F = \{F, A, C\}$.

Exercise 11.6 [Pitch - Distance, Chord Progressions]

On the basis of the Chord distance rule, [3] defines a distance function for chord sequences, called the Tonal Pitch Step Distance (TPSD). The central idea behind the TPSD is to compare the change of chordal distance to the tonic over time. This is done by calculating the chordal distance $d(x, y)$ between each chord y of the song and the triadic tonic chord x of the key of the song. Plotting the chordal distance against the time, results in a step function. The difference between two chord sequences can then be defined as the minimal area between the two step functions f and g over all possible horizontal shifts t of f over g (cp. Tables **??**, **??**). These shifts are cyclic, and to prevent longer sequences from yielding higher scores, the score is normalized by dividing it by the length of the shortest step function. Trivially, the TSPD can handle step functions of different length since the area between non-overlapping parts is always zero.

For the progressions in Exercise 11.5, determine the pairwise TPSD distances. Note that the progressions are 32 beats long, and that the triadic tonic chord x of the key of all songs is $F = \{F, A, C\}$. First, identify the step functions corresponding to the chord progressions I - V on the basis of the solutions of Exercise 11.5, and then calculate the corresponding TPSD distances for each pair of chord progressions.

Exercise 11.7 [Pitch - Chord clustering]

Apply hierarchical clustering ("complete" linkage) to the chord distances calculated in Exercise 11.5 (see Table **??**). This clusters the chords according to their distance to the triadic tonic chord x of the key of all songs $F = \{F, A, C\}$. How many clusters appear to be most sensible? Can you interpret the results?

Hint: Use package "dendextend" in R: Before plotting, specify the labels of the chords, e.g., as follows:

```
dend <- as.dendrogram(X.clust)
labels(dend) <- expression(F^"+",F^6,F/A,F^"j7",F^"07",G^7,G^"7b9",G^"-7",A^7,A^"-7",B^"b6",
B^"b7",B^"07",C,C^7,D^7,D^"-7",E^7/F)[order.dendrogram(dend)]
```

Exercise 11.8 [Timbre - Clustering vs. True Classes]

Consider the same situation as in Exercise 9.6. However, we only study the tones 4530 - 4640 with **55 guitar** and **56 piano tones**, and the features MFCC 1 in the 1st block ("mfcc_block1_1") and in the 2nd block ("mfcc_block2_1") of the tones. Discuss whether **Ward clustering** is able to distinguish **guitar** and **piano tones** based on these two blocks. Compare with the results in Example 11.2 in the book.

Chapter 12

Supervised Classification
(Claus Weihs and Tobias Glasmachers)

For classification in R, we recommend the package `mlR`. A tutorial for this package can be found on `https://mlr-org.github.io/mlr-tutorial/release/html/index.html`.

Basic Exercises

Exercise 12.1 [Linear Classification]

Which of the following learning machines yields a linear classifier: LDA, 1-nearest-neighbor, decision tree, random forest, SVM with linear kernel, SVM with Gaussian kernel?

Exercise 12.2 [Genre - Nonlinear Classification]

What does it mean to apply the "kernel trick" to a linear method? What is the benefit, say, if a genre classification problem requires a non-linear decision boundary?

Exercise 12.3 [Pitch - Nearest Neighbors]

We want to train a regression model for predicting the pitch of a tone using a tone data base covering multiple instruments. For each tone we have MFCC, spectral envelope and chroma features, where the variability of the chroma feature values is ten times larger than for the other features. Do you expect a 1-nearest-neighbor predictor (based on Euclidean distance in feature space) to perform well on this task? If so: why, if no, why not? Try to come up with a better distance function.

Exercise 12.4 [Timbre - Instruments]

Train a decision tree to tell guitar from piano tones. The data file svm.RData (see Apendix A.1) contains features of tones, the instruments are encoded as labels 0 (guitar) and 1 (piano).

Hint: The R package `rpart` provides a decision tree and functions to plot the tree.

Exercise 12.5 [Timbre - Random Forest, Instruments]

Consider the data set svm.RData consisting of MFCC, spectral envelope and chroma features of guitar and piano tones. Train a random forest to disambiguate the instruments. The random forest method is commonly applied for estimating the relevance of features by checking which features are used most prominently by the trees. This can be done for example with help of the Gini impurity index.

Hints:

- The R package `randomForest` provides a random forest predictor.
- The importance of the variables can be plotted with the function varImpPlot() from the R package `randomForest`.

Advanced Exercises

Exercise 12.6 [Genre - Nonlinear SVM]

Train an SVM for telling classic from other music, analog to the example throughout the chapter. Load the data set dataTrain.csv consisting of MFCC features of 4-second windows of music into R. Apply a Gaussian kernel SVM classifier. Tune the SVM parameter C and the kernel parameter γ with grid search in the range $C \in \{2^0, 2^1, \ldots, 2^{10}\}$ and $\gamma \in \{2^{-2}, 2^{-1}, \ldots, 2^3\}$ using cross-validation. Report the best parameters and the cross-validation performance.

Hints:

- The R package `e1071` includes the SVM classifier.
- The R package `mlr` provides the functions makeParamSet() to set the parameters, makeResampleDesc() for cross validation and tuneParams for grid search.

Exercise 12.7 [Genre - Model Selection]

Load the genre (classic vs. non-classic) classification data sets dataTrain.csv and dataTest.csv. Train k-nearest-neighbor classifiers for $k = 1$, $k = 10$, and $k = 100$ and compute the error rate on training and test set. For which values of k do you observe underfitting or overfitting?

Hint: The R package `class` provides a knn classifier.

Exercise 12.8 [Timbre - Feature Selection]

Train an LDA model on the task of Exercise 12.5 in four conditions: (i) with all features, (ii) only with the MFCC features, (iii) only the with spectral envelope features, and (iv) only the with the chroma features. Estimate the error rates using cross-validation. Interpret the result.

Hints: The R package `mlR` provides an LDA classifier as well as the function makeResampleDesc() for cross validation.

Chapter 13

Evaluation

(Igor Vatolkin and Claus Weihs)

Basic Exercises

Exercise 13.1 [Evaluation Measures]

Which groups of evaluation measures exist? Discuss the differences and provide examples.

Exercise 13.2 [Multi-Objective Evaluation]

Which three steps belong to design of evaluation? Which combinations of measures can be used for multi-objective evaluation?

Exercise 13.3 [Genre - Evaluation of Decision Trees]

Let us implement a simple evaluation scenario for two decision trees in MATLAB. For the binary classification task Pop/Rock, load the training set with `c1 = arff_loader('PopRock-TS120-FP1.arff');` and the test set with `c2 = arff_loader('PopRock-TAS120-FP1.arff');`. Note that the last two attributes correspond to track id and category and do not belong to features. Before the classification, create a vector for labels and map category *AG_Pop* to 1 and *NOT_AG_Pop* to 0. Create the first decision tree using the first half of the training set, and the second one using the complete training set with the help of `fitctree`. Implement the following functions with regard to Definition 13.8 and Equations 13.7, 13.8, 13.14:

```
function [tp,tn,fp,fn] = confMatrix(trueLabel,predictedLabel)
function rec = recall(trueLabel,predictedLabel)
function prec = precision(trueLabel,predictedLabel)
function bre = balancedError(trueLabel,predictedLabel)
```

Output the evaluation measures and visualize the trees.

Exercise 13.4 [Timbre - Instruments]

Consider the same situation as in Exercise 9.6. However, we only study **10 guitar** and **11 piano tones**, and only the one feature MFCC 1 in the first block ("mfcc_block1_1") with the following elements (1st value changed, 1st 10 elements guitar!):
X = (2.4 1.26 1.59 1.81 1.32 1.36 1.21 0.85 1.65 0.81 4.31 3.37 2.54 3.05 5.07 3.20 2.85 2.74 4.88 2.75 2.54).
Simulating 5 repetitions of 50%-subsampling leads to the following random drawings of 11 elements of X in the training sample and 10 elements in the test sample. The corresponding classes (class1 – class5) are shown underneath the sample values (permu1 – permu5).

```
training samples
permu1 3.37 5.07 2.74 2.40 3.05 0.85 1.26 2.54 1.21 4.31 2.85
class1 1    1    1    0    1    0    0    1    0    1    1
permu2 1.21 4.88 1.65 2.54 0.85 1.32 1.26 3.20 4.31 2.75 2.54
class2 0    1    0    1    0    0    0    1    1    1    1
permu3 3.20 2.85 1.81 2.54 3.05 1.26 1.32 1.59 1.36 3.37 0.85
```

```
class3 1    1    0    1    1    0    0    0    0    1    0
permu4 3.37 1.36 3.20 2.54 4.88 1.32 1.21 1.59 2.75 2.40 5.07
class4 1    0    1    1    1    0    0    0    1    0    1
permu5 0.85 3.37 2.74 1.21 2.54 1.81 2.85 2.75 1.59 1.26 4.88
class5 0    1    1    0    1    0    1    1    0    0    1

test samples
permu1 1.59 4.88 1.32 1.36 0.81 1.65 1.81 2.75 2.54 3.20
class1 0    1    0    0    0    0    0    1    1    1
permu2 2.40 3.37 3.05 5.07 1.59 1.81 1.36 2.85 0.81 2.74
class2 0    1    1    1    0    0    0    1    0    1
permu3 4.88 2.54 2.74 2.40 0.81 1.65 4.31 5.07 2.75 1.21
class3 1    1    1    0    0    0    1    1    1    0
permu4 1.81 3.05 1.65 2.85 0.85 2.74 1.26 2.54 0.81 4.31
class4 0    1    0    1    0    1    0    1    0    1
permu5 4.31 1.65 1.32 1.36 3.20 2.54 5.07 2.40 3.05 0.81
class5 1    0    0    0    1    1    1    0    1    0
```

In order to classify, calculate the means of the elements of the training sample of the two classes, and use the mean of the means as the decision border. Afterwards, predict the unchosen elements. Report the errors. Which elements are misclassified and why?

Note that software is not necessary for solving the exercise since all information is given in the text so that you could do the calculations by hand! If you to do not want to do all calculations, concentrate on permutations 2 and 4.

Advanced Exercises

Exercise 13.5 [Genre - Multi-Objective Evaluation of Decision Trees]

In this exercise we will examine the performance of decision trees trained with different training set sizes. Based on the classification scenario from Exercise 13.3, write the MATLAB code which trains trees for all possible training sets between 15 and 120 tracks. Estimate recall, precision, and balanced relative error, and plot them together with the corresponding regression line using `fit(x,y,'poly1')`. Which impact on evaluation measures can you observe for increasing training set sizes?

Estimate hypervolumes for all three pairs of evaluation measures and increasing training set sizes with the help of `getHypervolume([firstMeasure, secondMeasure], isMinFirst, isMinSecond)`. Set `isMinFirst` and `isMinSecond` to 1 when the measure is to be minimized and to 0 when the measure is to be maximized. Which training set sizes have the largest hypervolumes with regard to the combinations of evaluation measures?

Exercise 13.6 [Genre - Nested Resampling]

Implement nested resampling in MATLAB using both data sets from Exercise 13.5 and a decision tree classifier:

```
fitctree(trainingInstances,trainingLabels,'MaxNumSplits',splitNumber);
```

For clarity we distinguish between following sets: in the outer cross-validation loop 9/10 of all instances (music tracks) are used as *tuning set* and the remaining 1/10 as *test set*. In the inner loop, the tuning set is divided into *training set* (9/10 of the tuning set) and *validation set* to find the best parameter setting for classification tree (1/10 of the tuning set). For the hyperparameter tuning in the inner loop, vary the number of maximum splits from 1 to 10 and run 10-fold cross-validation to find the best number of maximum splits estimating mean balanced relative error. For the estimation of test errors during outer cross-validation, use the best maximal split size each time and estimate finally the mean test error. Before the first start of the outer and each inner loop, shuffle the order of instances using `randperm`.

Exercise 13.7 [Timbre - Instruments]

Consider the same situation as in Exercise 9.6. However, we only study the tones 4530 - 4640, including **55 guitar** and **56 piano tones**, and the features MFCC 1 in the first block ("mfcc_block1_1") and in the second block ("mfcc_block2_1") of the tones (cp. Exercise 11.8). Apply leave-one-out cross-validation

to evaluate Linear Discriminant Analysis. The posterior probabilities of the observations for, e.g., class 0 can be estimated by "number of class 0 predictions during cross-validation" / "number of observations". If this estimated posterior probability is < 0.5, then this observation is classifies as "class 1". This leads to an error if the true class would be 0. Illustrate the misclassified observations.

Hint: Apply option "CV" in function "lda".

Exercise 13.8 [Pitch - Testing]

For the data in Exercise 13.7, apply the McNemar test to compare the results of Linear Discriminant Analysis (lda) and Classification Trees (rpart). Are there significant differences on the significance level 5%?

Chapter 14

Feature Processing
(Igor Vatolkin)

Basic Exercises

Exercise 14.1 [Categorization of Feature Processing Methods]

Describe several criteria how we can group feature processing methods with regard to their impact on feature matrix dimensions. Name example methods.

Exercise 14.2 [Missing Values]

List several reasons why some time windows may correspond to missing or "not a number" feature values before processing. Discuss the difference between audio features and other feature sources. Name concrete examples if possible. How the problem of missing values can be handled? Is it possible that some values will be missing after feature processing?

Exercise 14.3 [Automatic Feature Construction]

Which is the motivation to apply automatic feature construction? How new features can be constructed from others? Provide a categorization of methods with examples. Which is the main challenge of the method?

Exercise 14.4 [Evaluation of Feature Processing]

How can the feature processing methods be evaluated? Discuss general and specific evaluation measures.

Advanced Exercises

Exercise 14.5 [Timbre - Normalization]

The file *2_14.arff* contains the spectral centroid extracted for the track 2.mp3 from "1000 Songs" database using 22,050 Hz as a sampling frequency and extraction frames of 512 samples without overlap.

1. Write the MATLAB code to normalize the values into an interval [0,1] according to min-max normalization method and plot the normalized values. Take into account that for some frames with too weak signal no centroid values could be extracted ("NaN"-values) and replace them with the median of the series. To load the centroid series, use a command

   ```
   c = arff_loader('2_14.arff');
   ```

2. Write the MATLAB code to normalize the series with softmax normalization and plot the normalized values.

Exercise 14.6 [Timbre - Feature Processing]

Using the centroid series after the replacement of missing values from Exercise 14.5, estimate the maximum possible number of classification windows of 5 s with 2.5 s overlap. Write the MATLAB code for the estimation of quartile statistics for classification windows of 5 s with 2.5 s overlap. To calculate original extraction windows of 512 samples which are contained in corresponding classification windows, estimate the boundaries in seconds. List the quartile statistics for the 10th classification window.

Exercise 14.7 [Timbre - Structural Complexity]

The file *2_39.arff* contains 13 MFCCs extracted for the track 2.mp3 from "1000 Songs" database using 22,050 Hz as a sampling frequency and extraction frames of 512 samples without overlap. Write a MATLAB function which estimates structural complexity, so that 5 s before and 5 s after the analysis frame are taken into account. Plot the result.

Some hints:

- After the ARFF table is loaded, convert it to the matrix using `cell2mat` function. Remove the last column which corresponds to a window number.
- After the estimation of MFCC matrix, normalize the values to [0,1] using min-max normalization method as in Exercise 14.5. Why the normalization is necessary for the estimation of structural complexity?
- Define a separate function with two input vectors for the calculation of Kullback-Leibler divergence

Exercise 14.8 [Timbre - Instruments]

Let us study the 1st MFCC for a set of **62 cello** and **30 flute tones** as in Section 14.5. We distinguish between frames from the attack interval and the release interval, and only the "middle" of these intervals is taken into account. This leads to an acceptable separation of the distributions of **MFCC 1** for the two instruments (see Figures 14.5 (d,e,f)). Let us try to reconstruct this in a simplified way for the attack interval.

Use the data from /Data/Feature_Processing/MFCCs, where you can find the MFCC features for several tones. The first column corresponds to the 1st MFCC. Let us assume, the attack part consists of the first 10 windows. First, consider the mean values over all windows (for each tone) and compute the densities for cello and flute, respectively. Second, consider the first 10 windows of each tone only. Which approach is better for separation?

Hint: For reading ARFF files in R, we recommend the package `foreign` with the function `read.arff()`. For reading the file names within a directory, you can use the standard R function *dir()*.

Chapter 15

Feature Selection
(Igor Vatolkin)

Basic Exercises

Exercise 15.1 [Irrelevance and Redundancy]

Explain the difference between irrelevant and redundant features. Why does it make sense to remove both irrelevant and redundant features from classification models?

Exercise 15.2 [Feature Selection]

Name four design steps for a feature selection method and discuss their properties. Provide examples.

Exercise 15.3 [Relevance Criteria]

Name the categories of relevance criteria. Discuss their properties and provide examples for related measures.

Exercise 15.4 [Multi-Objective Feature Selection]

Which combinations of evaluation criteria can be used for the multi-objective feature selection? Which criteria are not suited for simultaneous optimization? Is it possible to estimate automatically how well two or more criteria are suited for simultaneous optimization?

Advanced Exercises

Exercise 15.5 [Genre - Correlation-Based Relevance]

The file *PopRock-TrainingSet4Tracks.arff* contains 15 processed features for 4 music tracks from the training set of Example 13.2. Write the MATLAB code for the estimation of average correlation between features using `corrcoef` function. Sort the features based on their average correlation to remaining features. Answer the following questions:

- Which two features have the strongest positive correlation?
- Which two features have the strongest negative correlation?
- Which two features have the weakest correlation?
- Which feature has the weakest average absolute correlation to other features (being the best candidate to add to an empty feature set during forward selection)?
- Which feature has the strongest average absolute correlation to other features (being the best candidate to remove from the complete feature set for during backward selection)?

Some hints:

- After the ARFF table is loaded with `arff_loader`, remove the last two columns of the cell array which correspond to a track ID and a category, and convert it to the matrix using `cell2mat`.
- Following MATLAB functions are useful: `sort` for sorting of matrices and arrays (for a matrix M, it can be sorted with `[value,index] = sort(M,:)`); `ind2sub` for the identification of features to identify after the sorting.

Exercise 15.6 [Genre - MRMR]

For the data set from Exercise 15.5, write the MATLAB code for minimal-redundancy-maximal-relevance (MRMR) feature selection. As the redundancy criterion W_H, use the absolute average correlation with already selected features, and as the relevance criterion V_H, use the absolute correlation with the label. Add features during a forward search one by one in a loop according to the maximization of $W_H - V_H$ and display the sequence of features which were added.

Some hints:

- In the original data set, the label is described as a string: positive Pop examples as *'AG_Pop_20'* and negative as *'NOT_AG_Pop_20'*. To measure the correlation between a feature and a label, replace these strings with values 1 for Pop examples and 0 for others. `strcmp` can be used for the parsing of string labels.
- To estimate the absolute values of the covariance, apply `abs` to the feature matrix after `corrcoef`.

Exercise 15.7 [Genre - MRMR, Decision Tree]

Modify the MATLAB code from Exercise 15.6 for the evaluation of MRMR-selected feature sets using classification with decision tree classifier and estimate the cardinality of the optimal feature set. Load the data set of 15 features and 120 tracks from *'PopRock-TAS120.arff'* and use the first half for training and the second half for testing. As an evaluation measure, implement the relative classification error, cf. Equation (13.12). Plot the tree of the set with the smallest error.

Some hints:

- For the training of the tree, use `fitctree(trainingSet,labels)`.
- For the classification with the trained tree, use `predict(tree,testSet)`.
- For the visualization of the tree, use `view(tree,'Mode','Graph')`.

Exercise 15.8 [Genre - Multi-Objective Evaluation of MRMR]

- Update the MATLAB code from Exercise 15.7 for the evaluation of MRMR-selected feature sets with regard to two pairs of criteria: (1) feature rate and relative error; (2) recall and specificity. To implement the measures, see Equation (**??**) and Section 13.3.3. Output the measures for all trees.
- With the help of the function `nonDominated(values,isMinFirst,isMinSecond)`, estimate the non-dominated fronts of incomparable solutions in both evaluation cases (cf. also Section 10.4).
- Plot the area which is dominated by the non-dominated fronts. Estimate the tree with the largest hypervolume in both evaluation cases. Is the same tree "optimal" for both evaluation cases? To measure the hypervolume, use the function `getHypervolume(values,isMinFirst,isMinSecond)`. Which features are contained in the tree(s) with the largest hypervolume?

Chapter 16

Segmentation

(Nadja Bauer, Sebastian Krey, Uwe Ligges, Claus Weihs, and Igor Vatolkin)

Basic Exercises

Exercise 16.1 [Duration - F-Measure]

Let $(0, 0.5, 1, 1.5)^T$ s be a vector of true onset times. Consider the following three vectors of estimated onset times:

a) $(1.49)^T$ s,

b) $(0.04, 0.45, 1.04, 1.49)^T$ s,

c) $(0.25, 0.75, 1.25, 1.49)^T$ s.

Calculate the F-values for all three vectors by using an ± 25 ms and ± 50 ms tolerance interval, respectively. Interpret the values. Which shortcomings of the F-value can you identify?

Exercise 16.2 [Duration - Moving Threshold]

Discuss the effect of the moving thresholding function *ThreshFunction* (*mean* or *median*) in formula (16.5) on the estimated onset times. Particularly discuss the effect of the size $r_T - l_T$ of the thresholding window for the two moving thresholding functions.

Exercise 16.3 [Duration - Spectral Flux]

In Chapter 5 the following definition of the spectral flux feature (see Formula (5.12)) is given:

$$t_{\text{flux}}[\lambda] = \sum_{\mu=0}^{M/2} \left(|X[\lambda, \mu]| - |X[\lambda - 1, \mu]|\right)^2 .$$

However, in the most onset detection publications the filter based version is used (also without squaring):

$$t_{\text{fluxH}}[\lambda] = \sum_{\mu=0}^{M/2} H(|X[\lambda, \mu]| - |X[\lambda - 1, \mu]|)$$

with $H(x) = (x + |x|)/2$. Discuss the effect of the $H(x)$ filter on onset detection.

Exercise 16.4 [Timbre - Phases of Instrument Sound]

What are the typical phases of an instrumental sound?

Exercise 16.5 [Timbre - Distance Measure]

Which distance measure is usually used for the visualization of features like MFCCs and how is it defined?

Exercise 16.6 [Order Constraints]

What is the advantage of order constrained solutions in k-means clustering in comparison to traditional clustering approaches (k-means, hierarchical clustering) for the application in music signal analysis?

Advanced Exercises

Exercise 16.7 [Duration - Onset Detection]

In the files `Piano.wav` and `Flute.wav` monophonic piano and flute recordings are given, illustrated in the book in Figure 16.3. The sampling rate of the two recordings is 44100 Hz.

1. Read both wave files and plot their amplitude. Pay attention to a correct labeling of the x-axis (either samples or time). In R you can use, e.g., the commands `readWave` and `plot` of the `tuneR` package.

2. Compute the short time Fourier transform (STFT). Use $M = 2048$ samples as the window length. The windows should not be overlapping. What are the smallest and the highest frequencies for which the Fourier coefficients are calculated? In R you can use, e.g., the command `specgram` of the `signal` package.

3. For the two audio recordings, calculate and illustrate two onset detection features: spectral flux ($t_{\text{fluxH}}[\lambda]$ as defined in Exercise 16.3) and the difference of the absolute amplitude maxima in the neighboring windows:

$$t_{\text{amplMax}}[\lambda] = absAmplitMax[\lambda] - absAmplitMax[\lambda - 1].$$

4. In the files `trueFrames_piano.txt` and `trueFrames_flute.txt` the frame numbers of the true onset times are gives (for the window length of 2048 samples). Illustrate these frames in your plots, e.g., by adding vertical lines which mark the onset frames. Discuss the plots with respect to the performance of the two features on the given recordings.

Exercise 16.8 [Duration - Onset Detection]

The aim of this Exercise is to demonstrate the effort required for hand labeling of tone onsets. Files `guitar.wav` and `violin.wav` contain 3 seconds of solo guitar and violin pieces, respectively.

Try to find the true onset times and mark them as vertical lines in the associated amplitude plots. In the R language the following commands of `tuneR` package might be helpful: `readWave`, `extractWave`, `play` and `plot`. It is also advisable to use onset detection features (mentioned, e.g., in Exercise 16.4) for faster finding of the most promising signal frames.

Exercise 16.9 [Order Constraints Clustering]

What are the necessary steps to cluster a musical piece into segments like in example 16.7 of the book?

Exercise 16.10 [Clustering a Short Piece with R]

Use R to produce the following data:

```
library("tuneR")
fjnotes <- c(rep(c(-9, -7, -5, -9), 2), rep(c(-5, -4, -2), 2))+5
fjnotes2 <- c(rep(c(-9, -10, -9, -14), 2), rep(c(-9, -7, -5), 2))-7
fjlengths <- c(rep(1, 8), rep(c(1, 1, 2), 2)) / 2
fjtones <- mapply(sine, 440*2^(fjnotes/12), duration = fjlengths, xunit="time")
fjtones2 <- mapply(sine, 440*2^(fjnotes2/12), duration = fjlengths, xunit="time")
fj <- do.call(bind, lapply(fjtones, prepComb))
fj2 <- do.call(bind, lapply(fjtones2, prepComb))
size <- min(length(fj), length(fj2))
fjm <- 0.7*fj[1:size] + 0.3*fj2[1:size]
```

The Wave object `fjm` contains a mono signal of two parts (voices) of “Frère Jaques”.

1. Write code to find k clusters of this signal using a simple k-means approach
2. and decide for an appropriate value of k (try each the clustering several times to see whether it is stable).
3. Do you get a temporal order $1, \ldots, n$ into k clusters? Does it make more sense to use an algorithm that preserves the temporal order or is the simple k-means approach sufficient?

Chapter 17

Transcription
(Uwe Ligges and Claus Weihs)

Basic Exercises

Exercise 17.1 [Pitch - Estimation]

Show, how the fundamental frequencies in Exercise 2.1 can be identified analogue to Exercise 4.8 by means of the method in Equation (17.1) and the **Quinn method** (Equation (17.2)) instead of the original time series. Is there a difference?

Hints:

- `FF()` in library `tuneR` realizes Equation (17.1).
- Additionally, we have prepared the R functions `quinn`, which realizes Equation (17.2).

Exercise 17.2 [Pitch - Notation]

Given the concert a' (440 Hz), what is the music notation for the fundamental frequencies 110 Hz and 138.65 Hz?

Exercise 17.3 [Duration - Note value]

Given the onsets 250 ms and 750 ms for the two notes A and c#/db of the same length, and the tempo 60 beats per minute, i.e., 60 quarters a minute, and no pause between the notes, what note value do the two notes have?

Exercise 17.4 [Duration - Notation]

Given length of notes, e.g., two quarters, draw the two notes A and c#/db into a graphical scheme such as the one produced by function `quantplot()` in the R package `tuneR`.

Advanced Exercises

Exercise 17.5 [Separation - ICA]

Run the following R code:

```
library("tuneR")
fjnotes <- c(rep(c(-9, -7, -5, -9), 2), rep(c(-5, -4, -2), 2))+5
fjnotes2 <- c(rep(c(-9, -10, -9, -14), 2), rep(c(-9, -7, -5), 2))-7
fjlengths <- c(rep(1, 8), rep(c(1, 1, 2), 2)) / 2
fjtones <- mapply(sine, 440*2^(fjnotes/12), duration = fjlengths, xunit="time")
fjtones2 <- mapply(sine, 440*2^(fjnotes2/12), duration = fjlengths, xunit="time")
fj <- do.call(bind, lapply(fjtones, prepComb))
fj2 <- do.call(bind, lapply(fjtones2, prepComb))
size <- min(length(fj), length(fj2))
fjs <- stereo(0.7*fj[1:size] + 0.3*fj2[1:size], 0.3*fj[1:size] + 0.7*fj2[1:size])
```

Now you have generated a stereo Wave object in R package `tuneR` that contains the first 8 bars of a very artificial version of the song "Frère Jaques".

Listen to the sound of the `fjs` object and apply an ICA in order to separate the two parts (voices) from each other. Convert the result into some sound again and listen t it in order to verify if you have been successful.

Exercise 17.6 [Pitch - Estimation]

With the pre-processed outcome of the previous exercise, go on and try to estimate fundamental frequencies (on windows) of the first part (voice) of "Frère Jaques", convert them to notes and visualize the melody you have found.

If you could not find a solution to the previous excercise, just use the `fj` object from the code above to continue.

Exercise 17.7 [Duration - Quantization]

Now, find a quantization of the result and visualize it.

Exercise 17.8 [Pitch - Typesetting]

Finally, try to produce sheet music.

Chapter 18

Instrument Recognition

(Claus Weihs, Klaus Friedrichs, and Kerstin Wintersohl)

Chapter 18 – Basic Exercises

Exercise 18.1 [Timbre - Characteristics]

Consider the data set *inst_orig.txt* (see Appendix A.7) with standard original features (characteristics) which is described in section 18.4. Discuss which feature is best to distinguish violin and clarinet. Use scatterplots for visualization.

Hint: The R function `pairs()` produces a matrix of scatterplots.

Exercise 18.2 [Timbre - Characteristics]

Please repeat Exercise 18.1 for the auditory features of channels 1, 14, 27, 40 (data set *inst_AM.txt*, see Appendix A.7). Which features and which channels are the best ones?

Exercise 18.3 [Timbre - Classification]

Discuss the performance of the useful features, which you have identified in Exercise 18.1 to separate clarinet and violine tones by means of a random forest based on the original observations. Compare the result to the performance of a random forest with all standard features.

Hint: The R package `randomForest` provides a random forest predictor.

Exercise 18.4 [Timbre - Classification]

Please repeat Exercise 18.3 with the auditory features of Exercise 18.2. Which feature set performs better?

Advanced Exercises

Exercise 18.5 [Timbre - Feature Selection]

In this exercise, we use only the features of one channel of the auditory model to separate the five instruments. Use a linear SVM and discuss the performances of different channels.

Hint: The R package `e1071` provides an SVM predictor.

Exercise 18.6 [Timbre - Hyperparameter Tuning]

Please combine the original and the auditory features into one data set. Use a linear SVM and a random forest for classification. Tune the hyperparameters of the classification methods and compare the error rates by 10-fold cross-validation to the individual results listed in section 18.4.

Exercise 18.7 [Timbre - Feature Selection]

Please apply feature selection for the combined data set of Exercise 18.6 and a polynomial SVM with the kernel function (compare Chapter 12)

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + 1000)^2 \text{ and with } C = 10^{-5}. \tag{18.1}$$

Use group-based forward and backward selection (compare section 18.4.5.3) and compare the results to the error rates without feature selection.

Exercise 18.8 [Timbre - Hierarchical Taxonomy]

Please test the hierarchical taxonomies for the combined data set of Exercise 18.6 and compare the results to the flat taxonomy (compare section 18.4.5.3). Use the same SVM as in Exercise 18.7

Chapter 19

Chord Recognition
(Geoffroy Peeters and Johan Pauwels)

Basic Exercises

Exercise 19.1 [Pitch - Chord]

What are the notes composing a D augmented triad ? a D dominant 7 tetrad ?

Exercise 19.2 [Pitch / Timbre - Harmonic Frequencies]

What are the first 4 harmonic frequencies of the pitch d3 ? What are the closest pitches to these frequencies ?

Exercise 19.3 [Pitch - Chroma]

For the signal of the previous exercise, if all harmonics have the same amplitude equal to 1 what is the corresponding 12-dimensional chroma vector ?

Advanced Exercises

Exercise 19.4 [Pitch - Constant-Q-transform]

Develop a Matlab code to perform the constant-Q-transform of an audio signal (see part 19.3.2). We will consider the frequencies between c3 and c7. We will use a value $b = 1$ (b is the number of frequency bins per semitone), hence $Q = 16.81$. We will apply a Hamming analysis window. Apply this transform to the audio signal "chordpno.wav".

1. Create the constant-Q-transform cosine and sine basis.
2. Display the cosine and sine basis of the constant-Q-transform. The plot to obtain is indicated in Figure 19.1.
3. Perform the projection of the audio signal on the cosine and sine basis.
4. Display the resulting constant-Q-transform of the audio signal. The plot to obtain is indicated in Figure 19.2.

Exercise 19.5 [Pitch - Chord recognition]

Develop a Matlab code to estimate the chord labels of an audio-signal. We will only consider the 12 Major and 12 minor triads for our chord dictionary (see Table 19.1). The chord will be represented using a knowledge-driven approach (see part 19.4.1). The estimation will be done on a frame-basis using a one-minus-cosine distance (see part 19.5). Apply this estimation to the audio signal "chordpno.wav" and comment the results.

1. Define the chord dictionary for the Major and minor triads

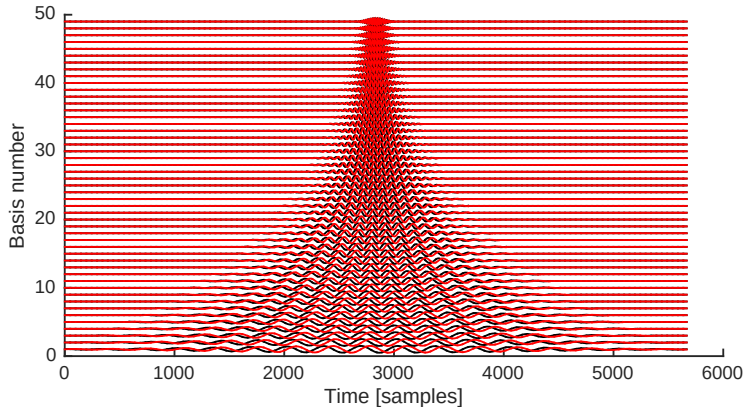

Figure 19.1: Representation of the cosine (black) and sine (red) basis of our constant-Q-transform.

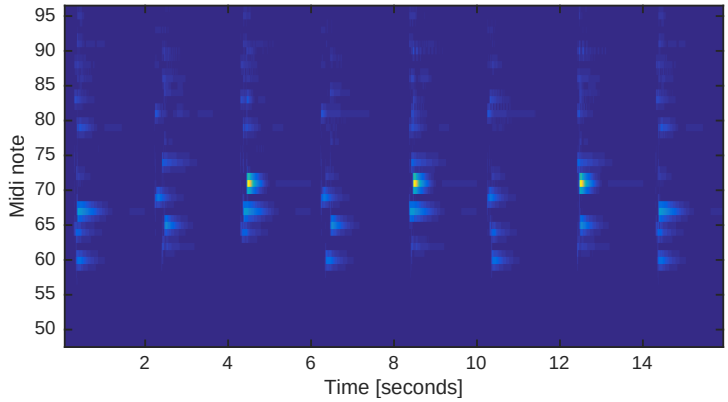

Figure 19.2: Resulting constant-Q-transform on the audio signal “chordpno.wav”.

2. Compute the chord representation in the 12 semitone pitch classes space of the chroma.
3. Display in matrix form (numberOfChord, numberOfChroma) the obtained chord dictionary values. The plot to obtain is indicated in Figure 19.3.

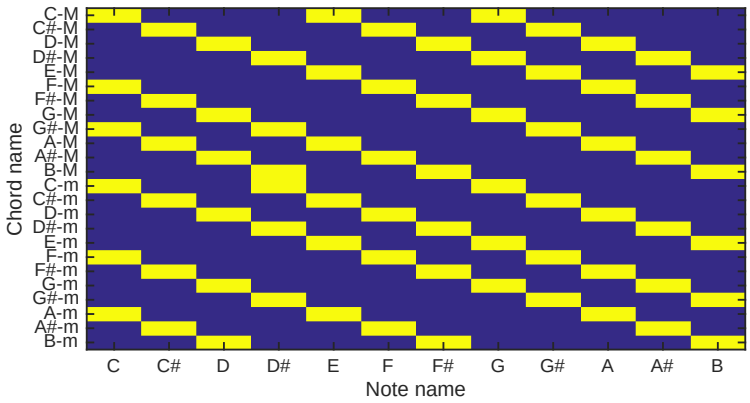

Figure 19.3: Display in matrix form (numberOfChord, numberOfChroma) of the Major and minor chord dictionary values.

4. Compute the chroma representation of the audio signal. This can be easily done starting from the constant-Q-transform representation of the audio signal.
5. At each time frame estimate the chord template that best represents the chroma vector using a one-minus-cosine-distance.
6. Display the estimated chords of the audio signal “chordpno.wav”. The plot to obtain is indicated in Figure 19.4.

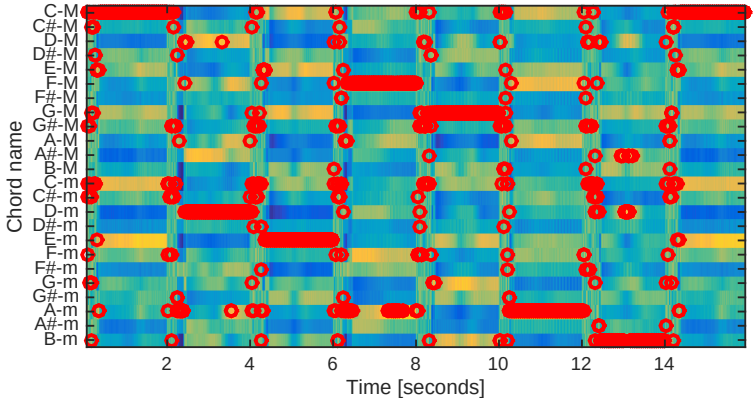

Figure 19.4: Estimated chords over time on the audio signal “chordpno.wav”.

Chapter 20

Tempo Estimation
(José R. Zapata)

Basic Exercises

Exercise 20.1 [Duration - Tempo]

What is Tempo in music?

Exercise 20.2 [Duration - Problems in Tempo Estimation]

List some reasons why it may be difficult to automatically estimate tempo in music?

Exercise 20.3 [Duration - Hierarchical Levels]

List the hierarchical levels of how rhythm metrical levels are divided starting with the fastest

Exercise 20.4 [Duration - Aim of Automatic Estimation]

Which is the aim of automatic rhythm estimation?

Exercise 20.5 [Duration - Feature List]

What is the purpose of the Feature list creation?

Exercise 20.6 [Duration - Tempo Induction]

What is the purpose of the tempo induction block in the tempo estimation process?

Exercise 20.7 [Duration - Applications]

List areas of research for which automatic rhythm estimation is relevant

Advanced Exercises

Implement the simple tempo estimation system proposed in the tempo chapter and download the tempo dataset from `http://www.music-ir.org/mirex/wiki/2006:Audio_Tempo_Extraction`

Exercise 20.8 [Duration - Tempo Estimation 1]

Plot the BPM Sum of all band autocorrelations of the file = train1.wav and locate the maximum values of the most prominent tempo values and compare with the ground truth provided in the dataset.

Exercise 20.9 [Duration - Tempo Estimation 2]

Plot the BPM Sum of all band autocorrelations of the file = train8.wav and locate the maximum values of the most prominent tempo values and compare with the ground truth provided in the dataset.

Exercise 20.10 [Duration - Tempo Estimation 3]

Plot the BPM Sum of all band autocorrelations of the file = train9.wav and locate the maximum values of the most prominent tempo values and compare with the ground truth provided in the dataset.

Exercise 20.11 [Duration - Tempo Estimation 4]

Plot the BPM Sum of all band autocorrelations of the file = train20.wav and and locate the maximum values of the most prominent tempo values and compare with the ground truth provided in the dataset.

Chapter 21

Emotions
(Günther Rötter and Igor Vatolkin)

Basic Exercises

Exercise 21.1 [Affective States]

Please name six affective states.

Exercise 21.2 [Emotions]

What are the basic emotions?

Exercise 21.3 [Theory of Emotions]

Describe the theory of emotions from George Mandler related to music.

Exercise 21.4 [Emotions and Music]

Does music elicit real emotions?

Advanced Exercises

Exercise 21.5 [Timbre - Characterization of Emotions]

The 1st MFCC is the most relevant feature for the identification of anger and sadness, the 2nd MFCC for fear (see Example 21.6). Visualize the distributions of MFCC 1 and MFCC 2 for anger, sadness, and fear using the data from Examples 21.1. – 21.4 (MFCCs_anger.csv, MFCCs_sadness.csv, MFCCs_fear.csv and MFCCs_joy.csv).

Exercise 21.6 [Categorical Emotion Recognition 1]

Reproduce the results of the Table 21.4 using MATLAB. First, load the features for 16 tracks with

```
[c,attr] = arff_loader('categorical.arff');
```

After the ARFF table is loaded, create the numerical feature matrix using `cell2mat` without the last two attributes (id and category). Then, estimate individual relevance of features for the recognition of four emotions anger, fear, joy, sadness with the help of Wilcoxon test (`ranksum` function). Output three features which are at highest individually relevant for the recognition of the emotions.

Exercise 21.7 [Categorical Emotion Recognition 2]

Extend the solution of Exercise 21.6 with feature processing which stores mean values of each feature for complete tracks, and not for classification windows of 4 seconds. To identify all windows which belong to

the same track, use the track id (second to last attribute in *categorical.arff*). Are the top three features which distinguish well between the emotions the same? Are the results reliable?

Exercise 21.8 [Dimensional Emotion Recognition]

Reproduce the results of the Table 21.5 using MATLAB. First, load the features for 100 tracks with

```
[c,attr] = arff_loader('dimensional.arff');
```

After the ARFF table is loaded, create the numerical feature matrix using `cell2mat` without the last attribute (id). Then, estimate R squared value for the regression between features and values of arousal and valence. To associate corresponding arousal and valence values with audio features, use two tables which map track id to mp3 name (*id2mp3.arff*) and mp3 name to arousal and valence (*static_annotations.csv*). In the last table, mp3 name is listed in the first column, arousal and valence in the second and the fourth column. For regression, use

```
[p,g] = fit(featMatrix,arousal,'poly1');
```

Output five features which explain at best arousal and five features which explain at best valence. Visualize the regression as in Figure 21.6.

Chapter 22

Similarity-Based Organization of Music Collections

(Sebastian Stober)

Basic Exercises

Exercise 22.1 [Album Similarity]

Derive a model for album similarity based on the approach for tracks described in this chapter!

Exercise 22.2 [Distance and Similarity]

Discuss why the duality of similarity and distance may be questioned from a psychological point of view as mentioned in Footnote 1!

Exercise 22.3 [Facet Weights]

Why is having negative facet weights problematic? (Cf. Equation 22.4)
Hint: This is, for instance, discussed by Cheng and Hüllermeier [2].

Exercise 22.4 [Distance Constraints]

How can the approach be changed to accommodate general relative distance constraints like *"a and b are more similar / less distant than u and v"*? How would you need to change the equations?

Advanced Exercises

The MagnaTagATune dataset, which is currently hosted at `http://mi.soi.city.ac.uk/blog/codeapps/the-magnatagatune-dataset`, contains a set of similarity annotations. This data was collected in a game where in each round, participants had to listen to three 30-seconds music clips and decide which one is the most dissimilar one. Parts of this dataset will be used in the following exercises.

Exercise 22.5 [Inferring Constraints]

1. Download the MagnaTagATune similarity data (comparisons_final.csv)! It contains a list of triplets with clip IDs and paths as well as the number of votes per clip from the similarity game.
2. Turn this data into a set of relative distance constraints!
3. The data contains many contradictions. Discuss possible reasons for this!
4. How could the game for collecting this data be changed to reduce or even avoid contradicting constraints?

Exercise 22.6 [Filtering Constraints]

Given the set of constraints obtained in Exercise 22.5, remove contradictions using the graph-based method described in Section 22.2.3. How many constraints are removed in each of the steps? What does this tell you about the data?

Exercise 22.7 [Pitch/Volume/Timbre/Duration - Audio Features & Distances]

1. Download the MagnaTagATune audio features (mp3_echonest_xml.zip) and unzip the file! This will result in one XML file per clip. The file names match with the path information provided with the similarity constraints appending ".xml" to each path.

2. For each clip referenced in the similarity constraints, extract the features listed in Table 22.1. The

feature	dim.	value description	distance measure
key	1	0 to 11 (one of the 12 keys) or -1 (none)	binary (exact match)
mode	1	0 (minor), 1 (major) or -1 (none)	binary (exact match)
loudness	1	overall value in decibel (dB)	absolute difference (L1)
tempo	1	in beats per minute (bpm)	absolute difference (L1)
time signature	1	3 to 7 ($\frac{3}{4}$ to $\frac{7}{4}$), 1 (complex), or -1 (none)	binary; $\delta(3,6) = 0.5$
pitch mean	12	dimensions correspond to pitch classes	Euclidean distance (L2)
pitch std. dev.	12	dimensions correspond to pitch classes	Euclidean distance (L2)
timbre mean	12	normalized timbre PCA coefficients	Euclidean distance (L2)
timbre std. dev.	12	normalized timbre PCA coefficients	Euclidean distance (L2)

Table 22.1: Features available for the MagnaTagATune dataset and respective distance measures. Top section: Globally extracted features. Bottom section: Aggregation of features extracted per segment.

   features pitch and timbre are given per segment. They need to be aggregated using the mean and standard deviation. Each feature will serve as a distance facet using the distance measures listed in the table.

3. For each distance constraint given by a triplet (s, a, b), compute the facet distance differences, i.e., the x_i in Equation 22.7 or the δw_i in Equation 22.12.

Exercise 22.8 [Pitch/Volume/Timbre/Duration - Learning Facet Weights]

Given the facet distance differences (one vector per distance constraint), learn facet weights that violate as few distance constraints as possible!

1. Apply the gradient descent learning scheme described in Section 22.2.4.1! Start with uniform weights and monitor the number of constraint violations after each update. Choose a suitable initial learning rate that results in reasonable step sizes. To reduce the learning rate progressively, you can multiply it with a factor between 0 and 1. Let the optimization run for a fixed number of iterations and keep track of the best solution. Optionally, you can aggregate the δw_i values from multiple violated constraints and then update the weights only once for such a mini-batch. This will result in much smoother weight updates.

2. For comparison, train a linear support vector machine (SVM) for the corresponding binary classification problem as described in Section 22.2.4.3! Extract the weights (feature coefficients) from the learned solution and check whether they are all non-negative (Equation 22.4). If some are negative, transform them accordingly and compare the number of constraint violations before and afterwards. Compare the result with the outcome of the gradient descent approach.

3. Which learning approach worked better? Which one was faster? Which one was easier to apply? Discuss possible reasons why it was impossible to satisfy all constraints!

Exercise 22.9 [Pitch/Volume/Timbre/Duration - Projections]

Apply multi-dimensional scaling (MDS) to project the clips from Exercise 22.7 into two-dimensional space. Compute one projection for each of the 9 distance facets and another one for a weighted combination of

all facets. (Either use uniform weights or the best solution you found in Exercise 22.8.) Visualize each projection, compute the respective trustworthiness and continuity scores, and compare the results.
Hint: Use pre-computed distance matrices as input to the MDS.

Chapter 23

Music Recommendation
(Dietmar Jannach and Geoffray Bonnin)

Basic Exercises

Exercise 23.1 [Basic Recommendation Techniques]

Describe the main ideas of collaborative filtering and content-based filtering techniques and their advantages and shortcomings.

Exercise 23.2 [Specifics of Music Recommendation]

In which ways can music recommendation problems be different from other typical application scenarios like the recommendation of shopping items on e-commerce sites?

Exercise 23.3 [Quality of Music Recommendation 1]

How can we evaluate the quality of music recommendations and why is this problem particularly challenging in academic settings?

Exercise 23.4 [Quality of Music Recommendation 2]

Discuss different factors that can influence the perceived quality of a recommendation list in the music domain.

Advanced Exercises

Exercise 23.5 [Playlists]

Download the playlists data file from the supporting website of the book. The file contains 1,000 playlists created and shared by users of Last.fm and has the following CSV fields:

`user_id,playlist_id,position,artist_name,track_name`.

The `position` field the represents the position of the track in the playlist.

Write an R script to convert the data into a file format which uses integers as IDs instead of artist and track names as follows.

- Each artist name must be mapped to a unique integer (artistID).
- Each track name must be mapped to a unique integer (trackID).
- Each line of the output file must contain one playlist. A playlist is represented as a sequence of pairs of the type "artistID:trackID", i.e., the artist and track IDs have to be separated by a colon character (":"). The playlist entries have to be separated by a space character.

The playlist with the following raw format:

```
6551544,599612,1,"The Beatles","Lucy in the Sky With Diamonds"
6551544,599612,2,"Radiohead","Creep"
6551544,599612,3,"Pink Martini","Sympathique"
6551544,599612,4,"Ella Fitzgerald","Let's Do It (Let's Fall In Love)"
6551544,599612,5,"No Doubt","Don't Speak"
6551544,599612,6,"Queen","Bohemian Rhapsody"
6551544,599612,7,"Queen","Another One Bites the Dust"
```

could be translated into the following representation:

```
1036:1503 876:547 843:2318 333:1393 769:640 869:351 869:192
```

Save the resulting data in a text file called *playlists-with-IDs.txt*. Also save your mappings from artist names and track names to the corresponding IDs to text files.

Some R hints:

- The content of the CSV file can be loaded using the function `read.csv`:
  `data <- read.csv("playlists.csv", encoding="UTF-8")`
- The list of distinct elements of the i^{th} column of the resulting data frame can be obtained using the functions `factor` and `levels`. For instance, the playlist IDs can be obtained by writing:
  `playlist_ids <- levels(factor(data[[2]]))`.
- Character vectors can be saved using the function `writeLines`:

```
fileConn <- file("artists-integer-ids.txt")
writeLines(artists, fileConn, useBytes=T)
close(fileConn)
```

Exercise 23.6 [Artist Co-Occurrences]

In the following three programming exercises, we will develop different music recommenders that generate continuations of playlists.

People love popular tracks. Program a baseline algorithm *PopRank* which simply recommends the most popular tracks. The algorithm should take the file *playlists-with-IDs.txt* of the previous exercise as an input and compute the 10 most popular tracks. The most popular tracks are those that appear most often in the playlist file.

Inspect the resulting playlist continuation by mapping the IDs back to artist and track names. Which are the most popular tracks in the dataset?

Some R hints:

- Some code to load the playlists and the ID mappings can be found in the file *data-loader.R*.
- Some helper functions to handle the mappings between the IDs and the artist and track names and to display the recommendations in a readable way can be found in the file *helpers.R*. The file also contains other functions that will be useful for the next exercises.
- In order to compute the occurrences of each track in the playlists, the function `table` can be used. Concatenate all playlists in one single vector and then apply the function on the resulting vector.
- The final recommendation list can then be obtained by sorting the tracks in decreasing order of the number of occurrences using the function `sort` and by selecting the 10 first elements.

Exercise 23.7 [Nearest Neighbors]

Our next playlisting algorithm will go a step further by taking into account a given playlist beginning additionally to the set of playlists. Develop a *k-Nearest-Neighbor* (kNN) algorithm as described in [1] and test it with some arbitrary playlists.

The kNN algorithm generally works as follows. It takes a given playlist as an input and compares it with all other existing playlists to determine a set of k playlists which are the most similar ones compared to the given playlist. The most similar playlists are called *neighbors*. The algorithm then

selects tracks appearing in these neighboring playlists as possible continuations of the given playlist. The underlying assumption is that if tracks frequently co-occurred in similar playlists they are also suited for the given playlist. To rank the possible next tracks, a *score* is computed for every track that appears in a neighboring playlist.

More formally, the algorithm should work as follows:

1. For each playlist p in the set of playlists
   (a) Compute the binary cosine similarity of p with the given playlist g:
   $$sim(p, g) = \frac{\|p \cap g\|}{\sqrt{\|p\|\|g\|}}$$
   (b) If the similarity is greater than 0, determine the tracks that are not present in the given playlist.
   (c) Update the score of each of these tracks by adding the similarity value to the previous score.
2. Return the top 10 tracks ordered by their score.

Inspect the resulting playlist continuations and assess their suitability for the given playlist.

Some R hints:

- The overlap between each playlist of the playlist dataset and the given playlist can be obtained using the function `intersect`.
- The input playlist beginning can be constructed using the function `addTrack` of the file *helpers.R*.

Exercise 23.8 [Co-occurance]

Our final algorithm uses a combination of popularity and co-occurrence patterns. It is called "Collocated Artists – Greatest Hits (CAGH)" and described in [1]. The general idea is to find *artist* co-occurrences. Let us assume that the given playlist for example contains tracks by The Beatles. If many playlists that contain tracks of The Beatles also contain tracks of The Rolling Stones, the CAGH algorithm will return (among others) the most popular tracks of The Rolling Stones.

The algorithm computes a score for each recommendable track and works as follows.

1. Count the occurrences of all the tracks of all the artists.
2. Compute the artist occurrences.
3. Compute all artist collocation values (i.e., the number of times each possible pair of artists appears in a same playlist)
4. For each artist a of the given playlist g:
   (a) Find the most similar artists to a in the set of playlists. The similarity between two artists a and b is the binary cosine:
   $$sim(a, b) = \frac{\sum_p (1_p(a) \cdot 1_p(b))}{\sqrt{\sum_p 1_p(a) \cdot \sum_p 1_p(b)}}$$
   with $1_p(a) = 1$ if a playlist p contains a and 0 otherwise.
   (b) Set the score of each of the tracks of each of the similar artists b as: track count times the computed similarity.
5. Return the top 10 tracks ranked by the computed score.

Again, inspect the resulting playlists and assess their suitability for the given set of test playlists.

Some R hints:

- The set of artists of a playlist can be obtained using the function `extractArtists` of the file *helpers.R*.
- The occurrences of tracks per artist can be stored as key-value pairs using lists:

```
key <- queen
value <- queensTrackCount
artistTrackCounts <- list()
artistTrackCounts[[ queen ]] <- queensTrackCount
```

In the code above, the variable `queen` contains the ID value of the artist Queen (e.g., 869). The variable `queensTrackCount` contains associations between track IDs and the number of occurrences of the corresponding track.

Exercise 23.9 [Quality of Playlist Continuations]

Read paper [1] and discuss the possible different ways of assessing the quality playlist continuations.

Chapter 24

Automatic Composition
(Maik Hester and Bileam Kümper)

Basic Exercises

Exercise 24.1 [Pitch - L-System]

Apply the rules in Section 24.3.1.1 (L-system) to the 2 notes {a} and {b}.

Hint: The generation of L-Systems can be done by the R function "LSys" provided in "functions".

Exercise 24.2 [Pitch - Mutation]

Expand the code for Basic Exercise 24.1 with rules from 24.2.2.3 (mutation) to compose a 2-note piece of music.

Hint: The generation of L-Systems and mutations can be done by the R functions "LSys" and "mutation" provided in "functions". Use the output of "LSys" as the input of "mutation".

Exercise 24.3 [Pitch - Weighted Random Composition]

Compose a whole tone melody of a given probability of intervals:

- diminished fifth down: 2/20
- major third down: 3/20
- major second down: 4/20
- unison: 2/20
- major second up: 4/20
- major third up: 3/20
- diminished fifth up: 2/20

Choose a starting note and a total length and let the computer compose the other notes.

Exercise 24.4 [Recombination]

Take a number of melodies (the more the better) in the same meter and key (transposed if necessary) and cut them into pieces, each one bar long. Mix the pieces and draw eight of them to get a new melody. Repeat the exercise ten times and write down the resulting melodies. Choose one that you (or your friends) like best. What makes a melody a "good" melody? Keep the previously cut pieces for Advanced Exercise 24.5.

Advanced Exercises

Exercise 24.5 [Composition Rules]

In order to improve the melodies in Basic Exercise 24.4 evaluate them by checking the following criteria:

- The last note should be the tonic of your key.
- The first note should be one of the tonic chord notes of your key.
- Bar 4 should end with a note from the dominant chord.

If you are not satisfied, yet, find more rules.

Exercise 24.6 [Dependent Probability Composition]

Expand the code for Basic Exercise 24.3 by applying a little Markov chain. For each interval let the probability of the next interval be as follows:

- diminished fifth down:
  - unison: 1/10
  - major second up: 1/10
  - major third up: 3/10
  - diminished fifth up: 6/10
- major third down:
  - unison: 1/10
  - major second up: 2/10
  - major third up: 3/10
  - diminished fifth up: 3/10
- major second down
  - major second down: 1/10
  - unison: 2/10
  - major second up: 3/10
  - major third up: 2/10
  - diminished fifth up: 2/10
- unison:
  - major third down: 1/10
  - major second down: 2/10
  - unison: 1/10
  - major second up: 3/10
  - major third up: 1/10
  - diminished fifth up: 2/10
- major second up:
  - major third down: 2/10
  - major second down: 2/10
  - unison: 2/10
  - major second up: 3/10
  - major third up: 1/10
- major third up:
  - diminished fifth down: 1/10

  - major second down: 3/10
  - unison: 2/10
  - major second up: 2/10
  - major third up: 1/10
  - diminished fifth up: 1/10
- diminished fifth up:
  - major third down: 3/10
  - major second down: 4/10
  - unison: 2/10
  - major second up: 1/10

Exercise 24.7 [Pitch - Evaluation]

Compare the pitches of your new melodies from Exercise 24.5 with those of one of your favorite songs. For the first note in each bar calculate the difference between the pitches in halftone steps. Sum up all the values and find the mean value. The melody with the smallest mean value should be closest to your favorite song. Is it the best one?

Exercise 24.8 [Live Automatic Sound Installation]

Design a program for a live automatic sound installation which delivers sound from a live audio stream or previously recorded files. The program should decide whether to play back a file from its archive or to record a new file from the audio stream. While the archive grows there should always be a certain probability for new recordings.

Chapter 25

Implementation Architectures
(Martin Botteck)

Exercises

Exercise 25.1 [Processing Chain]

What are the three main processing stages required to sort a set of music tracks into categories? Provide a short description of each stage's task!

Exercise 25.2 [Processing Chain]

Which of the stages listed above is least demanding in terms of processing power?

Exercise 25.3 [Technical Criteria]

Which other technical criteria besides processing power required are to be applied in order to valuate a potential implementation architecture?

Exercise 25.4 [Further Criteria]

Why is it not enough to search for a solution that constitutes an optimum according to the technical criteria listed above? List at least three further criteria!

Exercise 25.5 [Player Device Processing]

A mobile phone CPU running at 1 GHz and consuming $3,56 \cdot 10^{-10}$ mWh per clock cycle will be able to classify 400 songs/minute. Feature processing takes 3 minutes per song on average. The phone provides a battery holding 1.800 mAh at 3,6 Volts. How long does it take for a music collection of 4.000 songs to

1. extract and process the features needed?
2. classify the songs into a given category?
3. How many recharges of the battery will be needed?
4. How will these values change in case processing is performed by a background task consuming 5 percent of the CPU time?

Chapter 26

User Interaction
(Wolfgang Theimer)

Basic Exercises

Exercise 26.1 [Functions]

What are the functions of a music processing system?

Exercise 26.2 [Architecture]

Draw a top-level architecture of a music processing system which processes user input and communicates with other systems.

Exercise 26.3 [Audio input]

For what purpose can you use audio input in music processing systems?

Exercise 26.4 [Multi-modal interaction]

What are the advantages of multi-modal interaction?

Advanced Exercises

Exercise 26.5 [Query-by-humming]

Describe the query-by-humming concept for music retrieval.

Exercise 26.6 [Visual presentation]

How can a music collection be represented visually? Provide an example.

Exercise 26.7 [Context]

Define the term "context" in music interaction.

Exercise 26.8 [Implementation architectures]

Describe the impact of latency and user feedback for music processing systems.

Chapter 27

Hardware Architectures for Music Classification

(Ingo Schmädecke and Holger Blume)

Basic Exercises

Exercise 27.1 [Processor vs. ASIC]

List the advantages and disadvantages of general purpose processors (GPPs) over application-specific integrated circuits (ASICs).

Exercise 27.2 [FPGA]

What are the basic building blocks of an FPGA?

Exercise 27.3 [GPP vs. GPU]

A signal processing algorithm should be implemented either on a general purpose processor or a graphical processing unit. Which structural property of the algorithm would be beneficial for an implementation on a GPU and why? What is the limit of that property, if you are using a general purpose processor instead?

Exercise 27.4 [Parallel Processing Schemes]

Name two parallel processing schemes, which exploit data or instruction level parallelism to process multiple data points.

Advanced Exercises

Exercise 27.5 [Critical Path]

In order to determine the maximum reachable clock frequency of a custom logic circuit, the longest delay path (critical path) of the circuit has to be calculated. The following logic circuit should be analyzed: Given the following propagation delays t_p of the logic gates:

- $t_{p,AND} = 2\,\text{ns}$
- $t_{p,OR} = 2\,\text{ns}$
- $t_{p,XOR} = 3\,\text{ns}$

Determine the critical path through the circuitry and calculate the maximum clock frequency.

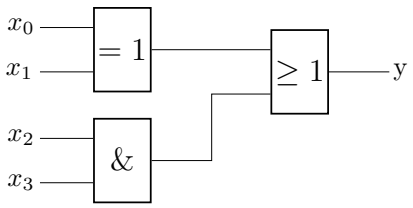

Figure 27.1: Custom logic circuit.

Exercise 27.6 [Quantitative Analysis of Data Path Architectures]

The following spectral difference square function (SDSF) calculates the squared difference between two audio spectrograms:

$$SDSF = \sum_{t=1}^{N} \sum_{f=1}^{M} [x(t,f) - y(t,f)]^2 \tag{27.1}$$

Therefore, the following operations are required for this calculation:

$$[M \times SUB + M \times MUL + (M-1) \times ADD] \times N \tag{27.2}$$

For the operations ADD, SUB and MUL the following cycle counts are defined for two target processors:

Operation	Cycles Processor 1	Cycles Processor 2
ADD	1	1
SUB	1	1
MUL	10	1

Table 27.1: Operation cycle counts for target processors 1 and 2

Calculate the required cycle counts for both target processors using equation 27.2 and table 27.1. Which processor is faster? What is the drawback with the faster processor?

Exercise 27.7 [Evaluation of Architectures using the ATE metric]

Three different architectures are available to classify a music data base of 1000 songs of equal length.

- The first architecture is a multi-core CPU, consisting of two cores running at 1 GHz. The area of each core is 435 mm^2. This multi-core CPU consumes about $3.56 \cdot 10^{-10}$ Wh per clock cycle and can classify 332 songs per minute.
- The second architecture is a GPU with 2000 cores consuming 100 watts. The whole data base is classified using this GPU in 161 seconds. The die size of the GPU is $5.1 \cdot 10^8$ um^2.
- The third architecture is an FPGA design. The area of the FPGA is 756 mm^2 and $5.2 \cdot 10^2$ J are consumed to classify the database. One song is classified within 0.1 seconds.

Determine the ATE cost for these three architectures, in order to evaluate the suitability of these architectures for music classification.

Exercise 27.8 [Pipeline]

1. In Fig. 27.2 a 5-stage instruction pipeline for a processor is shown. An assembler code for this processor is listed in Fig. 27.3. Identify all possible read after write (RAW) and write after read (WAR) conflicts.

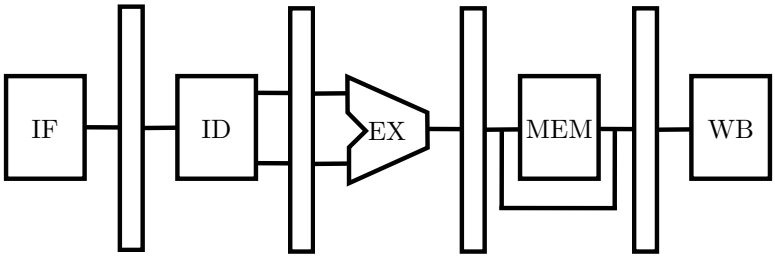

Figure 27.2: A 5-stage instruction pipeline architecture of a processor.

```
addi $t12, $t13, 20
addi $t13, $t14, 20
mul $t22, $s15, $s16
mul $t20, $s7, $s9
mul $t7, $t20, $s9
mul $t32, $s31, $s17
addi $t4, $t6, 20
addi $t3, $t1, 20
addi $t2, $t3, 20
```

Figure 27.3: Assembler code with conflicts.

2. The throughput of a digital signal processor can be increased by pipelining. The throughput of a processor with N pipeline stages is approximated using the following equation:

$$R = 1.1s^{-1} \cdot N \frac{5.3 \text{ ns} + 1.1 \text{ ns}}{5.3 \text{ ns} + N \cdot 1.1 \text{ ns}} \tag{27.3}$$

The area of the processor is increasing with the number of pipeline stages. This increase is approximated using the following equation:

$$A = (1.3 \text{ mm}^2 + N \cdot 0.15 \text{ mm}^2) \tag{27.4}$$

Determine the number of pipeline stages for the most efficient processor, by using the metric $E_{ff} = \frac{R}{A}$ to estimate the efficiency.

Bibliography

[1] G. Bonnin and D. Jannach. Automated Generation of Music Playlists: Survey and Experiments. *ACM Comput. Surv.*, 47(2):26:1–26:35, 2014.

[2] W. Cheng and E. Hüllermeier. Learning similarity functions from qualitative feedback. In *Proceedings of the 9th European Conference on Advances in Case-Based Reasoning (ECCBR'08)*, pages 120–134, Trier, Germany, 2008. Springer-Verlag.

[3] W. D. de Haas, R. C. Veltkamp, and F. Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. *Ismir*, pages 51–56, 2008.

Part II

Appendices

Chapter A

Appendix 1: Databases for Exercises

General

In this document we describe all databases used in the exercises. They can be downloaded from the book website: `http://sig-ma.de/music-data-analysis-book`.

List of Databases

1. *svmRData*: Characteristics of guitar and piano tones prepared for input in R.
2. *Genres*: Music and features for two genre classification tasks "classic against others" and "electronic against others"
3. *1000 Songs*: 744 copyright-free songs with annotated emotions and genres.
4. *Playlists*: Contains 1,000 playlists created and shared on Last.fm by music lovers. To be used for the advanced exercises on music recommendation (Chapter 23).
5. *Spectrum*: Spectra / periodograms of various real tones.
6. *Tempo Estimation*: Mirex tempo train dataset contains 20 audio excerpts with tempo ground truth annotation.
7. Instrument Features: Characteristics of instrument tones with noise due to accompanying instruments.

A.1 svm.RData

The **svm.RData** database (/Data/svm) includes characteristics of 4309 guitar tones and 1345 piano tones, the first 270 observations and the observations $546 - 4584$ representing guitar tones (Category = 0), the rest piano tones (Category = 1). The tones were taken from three databases, the McGill master samples collection on DVD [**?**], the RWC music data base [**?**], and the Iowa musical instrument samples [**?**]. The characteristics of these tones are divided into 5 blocks, the 1^{st} block represents the beginning of the tone, the last block the end. The characteristics enclose unwindowed MFCC ("mfcc_ungef"), for each block 13 MFCCs (e.g., "mfcc_block1_12"), for each block 25 Linear Prediction Codings LPCs (simplified spectral envelope)(e.g., "lpc_block3_17"), for each block 14 chroma characteristics (e.g., "chroma_block2_14"), and the absolute amplitude envelope in 132 windows (without blocks)(e.g., "huelle_31"). The data is prepared for input by R via

```
load("... (data repository) ...svm/svm.RData") .
```

```
  [1] "mfcc_ungef_1"     "mfcc_ungef_2"     "mfcc_ungef_3"
  [4] "mfcc_ungef_4"     "mfcc_ungef_5"     "mfcc_ungef_6"
  [7] "mfcc_ungef_7"     "mfcc_ungef_8"     "mfcc_ungef_9"
 [10] "mfcc_ungef_10"    "mfcc_ungef_11"    "mfcc_ungef_12"
 [13] "mfcc_ungef_13"    "mfcc_ungef_14"    "mfcc_ungef_15"
 [16] "mfcc_ungef_16"    "lpc_block1_1"     "lpc_block1_2"
 [19] "lpc_block1_3"     "lpc_block1_4"     "lpc_block1_5"
 [22] "lpc_block1_6"     "lpc_block1_7"     "lpc_block1_8"
 [25] "lpc_block1_9"     "lpc_block1_10"    "lpc_block1_11"
 [28] "lpc_block1_12"    "lpc_block1_13"    "lpc_block1_14"
 [31] "lpc_block1_15"    "lpc_block1_16"    "lpc_block1_17"
 [34] "lpc_block1_18"    "lpc_block1_19"    "lpc_block1_20"
 [37] "lpc_block1_21"    "lpc_block1_22"    "lpc_block1_23"
 [40] "lpc_block1_24"    "lpc_block1_25"    "lpc_block2_1"
 [43] "lpc_block2_2"     "lpc_block2_3"     "lpc_block2_4"
 [46] "lpc_block2_5"     "lpc_block2_6"     "lpc_block2_7"
 [49] "lpc_block2_8"     "lpc_block2_9"     "lpc_block2_10"
 [52] "lpc_block2_11"    "lpc_block2_12"    "lpc_block2_13"
 [55] "lpc_block2_14"    "lpc_block2_15"    "lpc_block2_16"
 [58] "lpc_block2_17"    "lpc_block2_18"    "lpc_block2_19"
 [61] "lpc_block2_20"    "lpc_block2_21"    "lpc_block2_22"
 [64] "lpc_block2_23"    "lpc_block2_24"    "lpc_block2_25"
 [67] "lpc_block3_1"     "lpc_block3_2"     "lpc_block3_3"
 [70] "lpc_block3_4"     "lpc_block3_5"     "lpc_block3_6"
 [73] "lpc_block3_7"     "lpc_block3_8"     "lpc_block3_9"
 [76] "lpc_block3_10"    "lpc_block3_11"    "lpc_block3_12"
 [79] "lpc_block3_13"    "lpc_block3_14"    "lpc_block3_15"
 [82] "lpc_block3_16"    "lpc_block3_17"    "lpc_block3_18"
 [85] "lpc_block3_19"    "lpc_block3_20"    "lpc_block3_21"
 [88] "lpc_block3_22"    "lpc_block3_23"    "lpc_block3_24"
 [91] "lpc_block3_25"    "lpc_block4_1"     "lpc_block4_2"
 [94] "lpc_block4_3"     "lpc_block4_4"     "lpc_block4_5"
 [97] "lpc_block4_6"     "lpc_block4_7"     "lpc_block4_8"
[100] "lpc_block4_9"     "lpc_block4_10"    "lpc_block4_11"
[103] "lpc_block4_12"    "lpc_block4_13"    "lpc_block4_14"
[106] "lpc_block4_15"    "lpc_block4_16"    "lpc_block4_17"
[109] "lpc_block4_18"    "lpc_block4_19"    "lpc_block4_20"
[112] "lpc_block4_21"    "lpc_block4_22"    "lpc_block4_23"
[115] "lpc_block4_24"    "lpc_block4_25"    "lpc_block5_1"
[118] "lpc_block5_2"     "lpc_block5_3"     "lpc_block5_4"
[121] "lpc_block5_5"     "lpc_block5_6"     "lpc_block5_7"
[124] "lpc_block5_8"     "lpc_block5_9"     "lpc_block5_10"
[127] "lpc_block5_11"    "lpc_block5_12"    "lpc_block5_13"
[130] "lpc_block5_14"    "lpc_block5_15"    "lpc_block5_16"
[133] "lpc_block5_17"    "lpc_block5_18"    "lpc_block5_19"
[136] "lpc_block5_20"    "lpc_block5_21"    "lpc_block5_22"
[139] "lpc_block5_23"    "lpc_block5_24"    "lpc_block5_25"
[142] "mfcc_block1_1"    "mfcc_block1_2"    "mfcc_block1_3"
```

```
[145] "mfcc_block1_4"    "mfcc_block1_5"    "mfcc_block1_6"
[148] "mfcc_block1_7"    "mfcc_block1_8"    "mfcc_block1_9"
[151] "mfcc_block1_10"   "mfcc_block1_11"   "mfcc_block1_12"
[154] "mfcc_block1_13"   "mfcc_block1_14"   "mfcc_block1_15"
[157] "mfcc_block1_16"   "mfcc_block2_1"    "mfcc_block2_2"
[160] "mfcc_block2_3"    "mfcc_block2_4"    "mfcc_block2_5"
[163] "mfcc_block2_6"    "mfcc_block2_7"    "mfcc_block2_8"
[166] "mfcc_block2_9"    "mfcc_block2_10"   "mfcc_block2_11"
[169] "mfcc_block2_12"   "mfcc_block2_13"   "mfcc_block2_14"
[172] "mfcc_block2_15"   "mfcc_block2_16"   "mfcc_block3_1"
[175] "mfcc_block3_2"    "mfcc_block3_3"    "mfcc_block3_4"
[178] "mfcc_block3_5"    "mfcc_block3_6"    "mfcc_block3_7"
[181] "mfcc_block3_8"    "mfcc_block3_9"    "mfcc_block3_10"
[184] "mfcc_block3_11"   "mfcc_block3_12"   "mfcc_block3_13"
[187] "mfcc_block3_14"   "mfcc_block3_15"   "mfcc_block3_16"
[190] "mfcc_block4_1"    "mfcc_block4_2"    "mfcc_block4_3"
[193] "mfcc_block4_4"    "mfcc_block4_5"    "mfcc_block4_6"
[196] "mfcc_block4_7"    "mfcc_block4_8"    "mfcc_block4_9"
[199] "mfcc_block4_10"   "mfcc_block4_11"   "mfcc_block4_12"
[202] "mfcc_block4_13"   "mfcc_block4_14"   "mfcc_block4_15"
[205] "mfcc_block4_16"   "mfcc_block5_1"    "mfcc_block5_2"
[208] "mfcc_block5_3"    "mfcc_block5_4"    "mfcc_block5_5"
[211] "mfcc_block5_6"    "mfcc_block5_7"    "mfcc_block5_8"
[214] "mfcc_block5_9"    "mfcc_block5_10"   "mfcc_block5_11"
[217] "mfcc_block5_12"   "mfcc_block5_13"   "mfcc_block5_14"
[220] "mfcc_block5_15"   "mfcc_block5_16"   "huelle_1"
[223] "huelle_2"         "huelle_3"         "huelle_4"
[226] "huelle_5"         "huelle_6"         "huelle_7"
[229] "huelle_8"         "huelle_9"         "huelle_10"
[232] "huelle_11"        "huelle_12"        "huelle_13"
[235] "huelle_14"        "huelle_15"        "huelle_16"
[238] "huelle_17"        "huelle_18"        "huelle_19"
[241] "huelle_20"        "huelle_21"        "huelle_22"
[244] "huelle_23"        "huelle_24"        "huelle_25"
[247] "huelle_26"        "huelle_27"        "huelle_28"
[250] "huelle_29"        "huelle_30"        "huelle_31"
[253] "huelle_32"        "huelle_33"        "huelle_34"
[256] "huelle_35"        "huelle_36"        "huelle_37"
[259] "huelle_38"        "huelle_39"        "huelle_40"
[262] "huelle_41"        "huelle_42"        "huelle_43"
[265] "huelle_44"        "huelle_45"        "huelle_46"
[268] "huelle_47"        "huelle_48"        "huelle_49"
[271] "huelle_50"        "huelle_51"        "huelle_52"
[274] "huelle_53"        "huelle_54"        "huelle_55"
[277] "huelle_56"        "huelle_57"        "huelle_58"
[280] "huelle_59"        "huelle_60"        "huelle_61"
[283] "huelle_62"        "huelle_63"        "huelle_64"
[286] "huelle_65"        "huelle_66"        "huelle_67"
[289] "huelle_68"        "huelle_69"        "huelle_70"
[292] "huelle_71"        "huelle_72"        "huelle_73"
[295] "huelle_74"        "huelle_75"        "huelle_76"
[298] "huelle_77"        "huelle_78"        "huelle_79"
[301] "huelle_80"        "huelle_81"        "huelle_82"
[304] "huelle_83"        "huelle_84"        "huelle_85"
[307] "huelle_86"        "huelle_87"        "huelle_88"
[310] "huelle_89"        "huelle_90"        "huelle_91"
[313] "huelle_92"        "huelle_93"        "huelle_94"
[316] "huelle_95"        "huelle_96"        "huelle_97"
[319] "huelle_98"        "huelle_99"        "huelle_100"
[322] "huelle_101"       "huelle_102"       "huelle_103"
[325] "huelle_104"       "huelle_105"       "huelle_106"
[328] "huelle_107"       "huelle_108"       "huelle_109"
[331] "huelle_110"       "huelle_111"       "huelle_112"
```

```
[334] "huelle_113"       "huelle_114"       "huelle_115"
[337] "huelle_116"       "huelle_117"       "huelle_118"
[340] "huelle_119"       "huelle_120"       "huelle_121"
[343] "huelle_122"       "huelle_123"       "huelle_124"
[346] "huelle_125"       "huelle_126"       "huelle_127"
[349] "huelle_128"       "huelle_129"       "huelle_130"
[352] "huelle_131"       "huelle_132"       "chroma_block1_1"
[355] "chroma_block1_2"  "chroma_block1_3"  "chroma_block1_4"
[358] "chroma_block1_5"  "chroma_block1_6"  "chroma_block1_7"
[361] "chroma_block1_8"  "chroma_block1_9"  "chroma_block1_10"
[364] "chroma_block1_11" "chroma_block1_12" "chroma_block1_13"
[367] "chroma_block1_14" "chroma_block2_1"  "chroma_block2_2"
[370] "chroma_block2_3"  "chroma_block2_4"  "chroma_block2_5"
[373] "chroma_block2_6"  "chroma_block2_7"  "chroma_block2_8"
[376] "chroma_block2_9"  "chroma_block2_10" "chroma_block2_11"
[379] "chroma_block2_12" "chroma_block2_13" "chroma_block2_14"
[382] "chroma_block3_1"  "chroma_block3_2"  "chroma_block3_3"
[385] "chroma_block3_4"  "chroma_block3_5"  "chroma_block3_6"
[388] "chroma_block3_7"  "chroma_block3_8"  "chroma_block3_9"
[391] "chroma_block3_10" "chroma_block3_11" "chroma_block3_12"
[394] "chroma_block3_13" "chroma_block3_14" "chroma_block4_1"
[397] "chroma_block4_2"  "chroma_block4_3"  "chroma_block4_4"
[400] "chroma_block4_5"  "chroma_block4_6"  "chroma_block4_7"
[403] "chroma_block4_8"  "chroma_block4_9"  "chroma_block4_10"
[406] "chroma_block4_11" "chroma_block4_12" "chroma_block4_13"
[409] "chroma_block4_14" "chroma_block5_1"  "chroma_block5_2"
[412] "chroma_block5_3"  "chroma_block5_4"  "chroma_block5_5"
[415] "chroma_block5_6"  "chroma_block5_7"  "chroma_block5_8"
[418] "chroma_block5_9"  "chroma_block5_10" "chroma_block5_11"
[421] "chroma_block5_12" "chroma_block5_13" "chroma_block5_14"
[424] "Category"
```

A.2 Genres

The genres database (/Data/Supervised_Classification) contains two genre classification tasks "classic against others" and "electronic against others".

The database contains the following files:

- dataTrain.csv: MFCC features of 2361 observations from "classic against others" (used in Chapter 12)
- dataTest.csv: MFCC features of 15387 observations from "classic against others" (used in Chapter 12)

A.3 1000 Songs

The data set (/Data/Feature_Processing) contains some features for one track of the original data base.

- The file *2_14.arff* contains the spectral centroid extracted for the track 2.mp3 from the database using 22,050 Hz as a sampling frequency and extraction frames of 512 samples without overlap.
- The file *2_39.arff* contains 13 MFCCs extracted for the track 2.mp3 from the database using 22,050 Hz as a sampling frequency and extraction frames of 512 samples without overlap.

URL for download of the original files: `http://cvml.unige.ch/databases/emoMusic/`.

A.4 Playlists

The comma-separated data file (/Data/Music_Recommendation) contains 1,000 playlists created and shared users of Last.fm. The file has the following format:

```
user_id,playlist_id,position,artist_name,track_name
```

where the `position` field represents the position of a track in a playlist.

An example playlist looks as follows:

```
6551544,599612,1,"The Beatles","Lucy in the Sky With Diamonds"
6551544,599612,2,"Radiohead","Creep"
6551544,599612,3,"Pink Martini","Sympathique"
6551544,599612,4,"Ella Fitzgerald","Let's Do It (Let's Fall In Love)"
6551544,599612,5,"No Doubt","Don't Speak"
6551544,599612,6,"Queen","Bohemian Rhapsody"
6551544,599612,7,"Queen","Another One Bites the Dust"
```

A.5 Spectrum

The database (Data/The_Musical_Signal/Spectrum) contains periodograms of real tones, the clarinet tones taken from the McGill database and the flute tones from the database "instrument samples". At the moment, there are two subdirectories, namely "Clarinet" and "Flute". The clarinet subdirectory includes the periodograms of the tones "D3" (146.832 Hz) till "D6" (1174.66 Hz)(see below). The flute subdirectory includes the periodograms of the tones "C4" (261.626 Hz) till "F#6" (1479.98 Hz)(see below). The data is prepared for input by R via

```
setwd("... (data repository) .../Spectrum")
```

Here is a list of the periodogram files in the database.

```
list.files(getwd())
[1] "Clarinet" "Flute"
list.files("Clarinet")
 [1] "bcl_A#3.csv" "bcl_A#4.csv" "bcl_A#5.csv" "bcl_A3.csv"  "bcl_A4.csv"
 [6] "bcl_A5.csv"  "bcl_B3.csv"  "bcl_B4.csv"  "bcl_B5.csv"  "bcl_C#4.csv"
[11] "bcl_C#5.csv" "bcl_C#6.csv" "bcl_C4.csv"  "bcl_C5.csv"  "bcl_C6.csv"
[16] "bcl_D#3.csv" "bcl_D#4.csv" "bcl_D#5.csv" "bcl_D3.csv"  "bcl_D4.csv"
[21] "bcl_D5.csv"  "bcl_D6.csv"  "bcl_E3.csv"  "bcl_E4.csv"  "bcl_E5.csv"
[26] "bcl_F#3.csv" "bcl_F#4.csv" "bcl_F#5.csv" "bcl_F3.csv"  "bcl_F4.csv"
[31] "bcl_F5.csv"  "bcl_G#3.csv" "bcl_G#4.csv" "bcl_G#5.csv" "bcl_G3.csv"
[36] "bcl_G4.csv"  "bcl_G5.csv"
list.files("Flute")
 [1] "fl_A4.csv"   "fl_A5.csv"   "fl_A#4.csv" "fl_B4.csv"   "fl_B5.csv"
 [6] "fl_C4.csv"   "fl_C5.csv"   "fl_C6.csv"   "fl_C#4.csv" "fl_C#5.csv"
[11] "fl_C#6.csv" "fl_D4.csv"   "fl_D5.csv"   "fl_D6.csv"   "fl_D#4.csv"
[16] "fl_D#5.csv" "fl_D#6.csv" "fl_E4.csv"   "fl_E5.csv"   "fl_E6.csv"
[21] "fl_F4.csv"   "fl_F5.csv"   "fl_F6.csv"   "fl_F#4.csv" "fl_F#5.csv"
[26] "fl_F#6.csv" "fl_G4.csv"   "fl_G5.csv"   "fl_G#4.csv" "fl_G#5.csv"
```

A.6 Tempo estimation

The mirex tempo train dataset contains 20 audio excerpts with tempo ground truth annotation.
URL for download: `http://www.music-ir.org/mirex/wiki/2006:Audio_Tempo_Extraction`.

A.7 Instrument Features

The database (/Data/Instrument_Recognition/inst_features) contains two feature sets with characteristics calculated by means of the MATLAB MIR-toolbox from music pieces which are temporally segmented w.r.t. the onsets of the melody tones. This means one observation corresponds to one melody tone with additional noise from accomponying instruments. The first feature set (instruments_original.txt) consists of features, which is derived from WAVE data directly. The second feature set (instruments_AM.txt) is derived by pre-processing the WAVE data using an auditory model. Both feature sets consist of 850 observations of 5 instruments (170 observations for each one). The first feature set contains 21 features and the target variable "Instrument". The second feature set contains $21 \cdot 40 = 840$ features and the target variable "Instrument". See Section 18.4 for further information. The data is prepared for input by R via

`data_orig <- read.table("... (data repository) ...inst_features/inst_orig.txt", header=T)` and
`data_AM <- read.table("... (data repository) ...inst_features/inst_AM.txt", header=T)`.

In R you can use functions like summary() or head() to get more information about the data sets. For example:

```
> summary(data_orig)
   mean_flux          sd.flux.           rms              lowenergy         rolloff
 Min.   : 0.0001   Min.   : 0.000   Min.   :0.000032   Min.   :0.1333   Min.   : 1330
 1st Qu.: 8.4050   1st Qu.: 1.620   1st Qu.:0.021383   1st Qu.:0.4000   1st Qu.: 2772
 Median :10.9810   Median : 2.643   Median :0.029542   Median :0.5000   Median : 3138
 Mean   :13.2769   Mean   : 3.808   Mean   :0.033780   Mean   :0.4946   Mean   : 3567
 3rd Qu.:16.3065   3rd Qu.: 4.978   3rd Qu.:0.042380   3rd Qu.:0.5789   3rd Qu.: 3938
 Max.   :67.0336   Max.   :30.919   Max.   :0.159860   Max.   :0.8333   Max.   :13231
   brightness      irregularity       entropy           mfcc1             mfcc2
 Min.   :0.1366   Min.   :0.3608   Min.   :0.3958   Min.   :-1.230   Min.   :-3.2051
 1st Qu.:0.4267   1st Qu.:0.9067   1st Qu.:0.6689   1st Qu.: 2.025   1st Qu.:-1.1181
 Median :0.5012   Median :1.1273   Median :0.7055   Median : 2.450   Median :-0.8543
 Mean   :0.5038   Mean   :1.1317   Mean   :0.6969   Mean   : 2.433   Mean   :-0.8246
 3rd Qu.:0.5795   3rd Qu.:1.4035   3rd Qu.:0.7362   3rd Qu.: 2.896   3rd Qu.:-0.5230
 Max.   :0.9851   Max.   :1.9149   Max.   :0.8878   Max.   : 4.895   Max.   : 1.5943
 mfcc3             mfcc4               mfcc5              mfcc6
 Min.   :-0.6602   Min.   :-1.297927   Min.   :-1.05026   Min.   :-1.57832
 1st Qu.: 0.8172   1st Qu.:-0.166017   1st Qu.:-0.34430   1st Qu.:-0.43302
 Median : 1.0727   Median : 0.062583   Median :-0.19446   Median :-0.23899
 Mean   : 1.0229   Mean   : 0.005945   Mean   :-0.19398   Mean   :-0.27408
 3rd Qu.: 1.3119   3rd Qu.: 0.244153   3rd Qu.:-0.04232   3rd Qu.:-0.07095
 Max.   : 2.7948   Max.   : 0.642582   Max.   : 0.95208   Max.   : 0.56779
 mfcc7               mfcc8              mfcc9             mfcc10
 Min.   :-1.251251   Min.   :-1.61326   Min.   :-1.3870   Min.   :-1.47283
 1st Qu.:-0.179461   1st Qu.:-0.22774   1st Qu.:-0.1137   1st Qu.:-0.21855
 Median : 0.043664   Median :-0.04636   Median : 0.1634   Median : 0.08024
 Mean   :-0.005631   Mean   :-0.02966   Mean   : 0.1441   Mean   : 0.01726
 3rd Qu.: 0.225765   3rd Qu.: 0.15962   3rd Qu.: 0.4147   3rd Qu.: 0.33581
 Max.   : 1.013437   Max.   : 1.33007   Max.   : 1.5591   Max.   : 1.20021
 mfcc11            mfcc12            mfcc13           Instrument
 Min.   :-1.5087   Min.   :-1.6512   Min.   :-1.0076   flute   :170
 1st Qu.:-0.3497   1st Qu.:-0.1393   1st Qu.:-0.2269   clarinet:170
 Median :-0.0706   Median : 0.1046   Median : 0.0307   oboe    :170
 Mean   :-0.0634   Mean   : 0.1197   Mean   : 0.0482   trumpet :170
 3rd Qu.: 0.2143   3rd Qu.: 0.3655   3rd Qu.: 0.2982   violin  :170
 Max.   : 1.5199   Max.   : 1.5448   Max.   : 1.3177
```

Chapter B

R- and Matlab-Functions for Exercises

General

In what follows all R- and Matlab-Functions prepared for the exercises will be described so that the reader has an idea of their aim and their input and output, and is able to study the code.

List of Functions

1. ACF: Autocorrelation function in R in order to estimate the fundamental frequency
2. YIN: Mean squared difference function in R in order to estimate the fundamental frequency
3. Ceps: Cepstrum in R in order to estimate the fundamental frequency
4. quinn: Quinn method in R in order to estimate the fundamental frequency
5. histogram: R function, plots two different histograms in one plot
6. scatterplot: R function, plot two different scatterplots in one plot
7. amplMax: R function, calculates the feature "amplitude maximum"
8. specFlux: R function, calculates the feature "spectral flux"
9. Lsys: R function to compose a 2-note piece of music after the rules of the L-System
10. mutation: R function to compose a 2-note piece of music after the rules of mutation
11. plot_spectrum: R function, plots the frequency spectrum

B.1 ACF

The function **ACF** calculates the autocorrelation function in R in order to estimate the fundamental frequency. You can specify the following input parameters:

w: object of class Wave (prepared by tuneR),

N: number of signal samples [default: 1000],

fmin: minimum frequency [50],

fmax: maximum frequency [2000],

plot: plot parameter [FALSE]

Output is the fundamental frequency f0.
The function can be called in R, e.g., via

```
ACF(W1, plot = TRUE).
```

The code of the function looks as follows:

```
ACF <- function(w, N = 1000, fmin = 50, fmax = 2000, plot = FALSE){
  x <- w@left # signal time series
  sr <- w@samp.rate # sample rate
  n <- length(x)/2
  a <- rep(0,n)
  tau <- 1:n
  for(i in 1:n){
    # equation 4.43
    a[i] <- sum(x[1:N]*x[(1+tau[i]):(N+tau[i])])
  }
  # find peaks in ACF:
  peak_idx <- which(diff(sign(diff(a)))==-2)+1
  idx_max <- round(sr/fmin) # no frequencies smaller than fmin
  peak_max <- which(peak_idx < idx_max)[length(which(peak_idx < idx_max))]
  idx_min <- round(sr/fmax) # no frequencies larger than fmax
  peak_min <- which(peak_idx > idx_min)[1]
  peaks <- peak_idx[peak_min:peak_max]
  # find first maximum of a:
  m <- max(a[peaks])
  idx <- abs(a[peaks] - m) < 1e-8
  idx <- which(idx)
  idx <- peaks[idx[1]] # index of fundamental frequency
  f0 <- sr / idx # fundamental frequency
  if(plot){
    l <- round(max(a))
    plot(a[1:(n/2)], type = "l", xlab = "Lag", ylab = "ACF")
    abline(0,0)
    lines(x = rep(idx, l*2+1), y = -l:l, col = "red")
  }
  return(f0)
}
```

B.2 YIN

The function **YIN** calculates the mean squared difference in R in order to estimate the fundamental frequency. You can specify the following input parameters:

w: object of class Wave (prepared by tuneR),

N: number of signal samples [default: 1000],

fmin: minimum frequency [50],

fmax: maximum frequency [2000],

plot: plot parameter [FALSE]

Output is the fundamental frequency f0.
The function can be called in R, e.g., via

```
YIN(W1, plot = TRUE).
```

The code of the function looks as follows:

```
YIN <- function(w, N = 1000, fmin = 50, fmax = 2000, plot = FALSE){
  x <- w@left # signal time series
  sr <- w@samp.rate # sample rate
  n <- length(x)/2
  d <- rep(0,n)
  tau <- 1:n
  for(i in 1:n){
    # equation 4.44
    d[i]<- sum((x[1:N] - x[(1+tau[i]):(N+tau[i])])^2)
  }
  # find local minimum in mean-squared difference:
  peak_idx <- which(diff(sign(diff(d)))==2)+1
  idx_max <- round(sr/fmin) # no frequencies smaller than fmin
  peak_max <- which(peak_idx < idx_max)[length(which(peak_idx < idx_max))]
  idx_min <- round(sr/fmax) # no frequencies larger than fmax
  peak_min <- which(peak_idx > idx_min)[1]
  peaks <- peak_idx[peak_min:peak_max]
  # find first minimum of d:
  m <- min(d[peaks])
  idx <- abs(d[peaks] - m) < 1e-8
  idx <- which(idx)
  idx <- peaks[idx[1]] # index of fundamental frequency
  f0 <- sr / idx # fundamental frequency
  if(plot){
    l <- round(max(d))
    plot(d, type = "l", xlab = "Lag", ylab = "mean-squared difference")
    abline(0,0)
    lines(x = rep(idx, l*2+1), y = -l:l, col = "red")
  }
  return(f0)
}
```

B.3 Ceps

The function **Ceps** calculates the cepstrum in R in order to estimate the fundamental frequency. You can specify the following input parameters:

w: object of class Wave (prepared by tuneR),

fmin: minimum frequency [50],

fmax: maximum frequency [2000],

plot: plot parameter [FALSE]

Output is the fundamental frequency f0.
The function can be called in R, e.g., via

```
Ceps(W1, plot = TRUE).
```

The code of the function looks as follows:

```
Ceps <- function(w,fmin = 50, fmax = 2000, plot = FALSE){
  x <- w@left # signal time series
  sr <- w@samp.rate # sample rate
  # compute cepstrum:
  s <- fft(x, inverse = FALSE)
  l <- log(abs(s))
  c <- Re(fft(l, inverse = TRUE))
  # find peaks in cepstrum:
  peak_idx <- which(diff(sign(diff(c)))==-2)+1
  idx_max <- round(sr/fmin) # no frequencies smaller than fmin
  peak_max <- which(peak_idx < idx_max)[length(which(peak_idx < idx_max))]
  idx_min <- round(sr/fmax) # no frequencies larger than fmax
  peak_min <- which(peak_idx > idx_min)[1]
  peaks <- peak_idx[peak_min:peak_max]
  # find first maximum of c:
  m <- max(c[peaks])
  idx <- abs(c[peaks] - m) < 1e-8
  idx <- which(idx)
  idx <- peaks[idx[1]] # index of fundamental frequency
  f0 <- sr / idx # fundamental frequency
  if(plot){
    l <- round(max(c))
    plot(c[2:length(c)], type = "l", xlab = "Quefrency (s)", ylab = "Cepstrum")
    abline(0,0)
    lines(x = rep(idx, l*2+1), y = -l:l, col = "red")
  }
  return(f0)
}
```

B.4 quinn

The function **quinn** realizes the Quinn method in order to estimate the fundamental frequency . You can specify the following input parameters:

oobj: object of class Wave (prepared by tuneR),

Output is the fundamental frequency f0.
The function can be called in R, e.g., via

```
quinn(W1).
```

The code of the function looks as follows:

```
quinn <- function(oobj){
  fftres <- fft(oobj@left)[-1]
  peak.2 <- which.max(abs(fftres[1:round(length(fftres)/2)]))
  alpha1 <- Re(fftres[peak.2-1]/fftres[peak.2])
  alpha2 <- Re(fftres[peak.2+1]/fftres[peak.2])
  delta1 <- alpha1/(1-alpha1)
  delta2 <- alpha2/(1-alpha2)
  delta <- if(delta1 > 0 && delta2 > 0) delta2 else delta1
  f0 <- (peak.2 + delta) * oobj@samp.rate/length(oobj@left)
  return(f0)
}
```

B.5 histogram

The function **histogram** plots histograms of two different samples together into one plot. You can specify, whether an approximated normal distribution should be plotted or not. You can specify the following input parameters:

- vec1: Vector of the first sample
- vec2: vector of the second sample
- main, xlab, ylab: graphical parameters, caption and labels for the axis
- norm: normal distribution [FALSE]

Output is a plot with two histograms.
The function can be called in R, e.g., via

```
histogram(M1[,1], M[,1], main = "MFCC1: Pure sine (red) vs. sine with overtones (blue)",
xlab = "MFCC", ylab = "density")
```

The code of the function looks as follows:

```
histogram <- function(vec1, vec2, main, xlab, ylab, norm = FALSE){
  m1 <- mean(vec1)
  m2 <- mean(vec2)
  s1 <- sd(vec1)
  s2 <- sd(vec2)
  h1 <- hist(vec1, plot = FALSE)
  h2 <- hist(vec2, plot = FALSE)
  d1 <- h1$density
  d2 <- h2$density
  b1 <- h1$breaks
  b2 <- h2$breaks
  ylim <- c(0, min(ceiling(d1), ceiling(d2)))
  xlim <- c(min(min(b1), min(b2)), max(max(b1), max(b2)))
  plot(h1, freq = FALSE, xlim = xlim, ylim = ylim, col = rgb(1,0,0,0.5),
       xlab = xlab, ylab = ylab, main = main, cex.axis = 1.5, cex.main = 1.5, cex.lab = 1.5)
  plot(h2, freq = FALSE, xlim = xlim, ylim = ylim, col = rgb(0,0,1,0.5),
       add = TRUE)
  if(norm){
    x <- seq(xlim[1], xlim[2], 0.01)
    curve(dnorm(x, mean = m1, sd = s1), add = TRUE, lwd = 3, lty = 2, col = rgb(1,0,0))
    curve(dnorm(x, mean = m2, sd = s2), add = TRUE, lwd = 3, lty = 2, col = rgb(0,0,1))
  }
}
```

B.6 scatterplot

The function **scatterplot** plots scatterplots of two different samples together into one plot. You can specify the following input parameters:

- x: Vector of the first sample
- y: vector of the second sample
- main, xlab, ylab: graphical parameters, caption and labels for the axis

Output is a plot with two scatterplots.
The function can be called in R, e.g., via

```
scatterplot(M1, M, "Pure sine (red) vs. sine with overtones (blue)", xlab = "MFCC1",
ylab = "MFCC2")
```

The code of the function looks as follows:

```
scatterplot <- function(x, y, main, xlab, ylab){
  xlim <- c(min(min(x[,1]), min(y[,1])), max(max(x[,1]), max(y[,1])))
  ylim <- c(min(min(x[,2]), min(y[,2])), max(max(x[,2]), max(y[,2])))
  plot(x, xlim = xlim, ylim = ylim, main = main, xlab = xlab, ylab = ylab,
       cex.axis = 1.5, cex.main = 1.5, cex.lab = 1.5, col = rgb(1,0,0,0.5))
  points(y, col = rgb(0,0,1,0.5))
}
```

B.7 amplMax

The function **amplMax** is implemented in R and calculates the 'amplitude maximum' feature (maximum of the absolute amplitude in each signal window). You can specify the following input parameters:

wave: object of class Wave (prepared by tuneR),

N: number of signal samples [default: 2048],

Output is the 'amplitude maximum' feature.

```
amplMax <- function(wave, N = 2048)
{
  offset <- seq(1, length(wave@left) - N, by = N)
  Apml.matrix <- matrix(0, N, length(offset))

  # i-th column of the matrix Apml.matrix is the signal amplitude of the i-th frame
  for (i in seq_along(offset)) {
    Apml.matrix[1:N, i] <- wave@left[offset[i]:(offset[i] + N - 1)]
  }
  Max.amplit.wind <- apply(Apml.matrix, 2, function(x) max(abs(x)))
  AmplitFeature <- diff(c(0, Max.amplit.wind))
  return(AmplitFeature)
}
```

B.8 specFlux

The function **specFlux** is implemented in R and calculates the 'spectral flux' feature (for each signal window: sum of positiv changes of the spectrogram compared to the previous window). You can specify the following input parameter:

- spectrogram: output of specgram function (signal R package).

Output is the 'spectral flux' feature.

```
specFlux <- function(spectrogram)
{
  S <- spectrogram$S #complex numbers
  n <- dim(S)[2]
  # the filter H
  H <- function(x){
    (x+abs(x))/2
  }
  SF <- numeric(n)

  for (i in 2:n){
    SF[i] <- sum( H(abs(S[,i])-abs(S[,(i-1)])) )
  }
  return(SF) # thus, the feature vector has the same no. of entries as the no. of windows
}
```

B.9 LSys

The function **Lsys** composes a 2-note piece of music following the rules of the L-System. You can specify the following input parameters:

- a, b: the two notes
- n: number of iterations
- Anfang: beginning of the music piece

Output is a sequence of the two notes a and b
The function can be called in R, e.g., via

```
LSys("a", "b", 5, Anfang = "a")
LSys(110, 138.59, 5, 110)
```

The code of the function looks as follows:

```
LSys <- function(a, b, n, Anfang){
  vec <- Anfang
  while(n != 0){
    l <- length(vec)
    i <- 1
    while(i <= l){
      if(vec[i] == a){
        vec[i] <- b
        i <- i+1
      }
      else{
        length(vec) <- l + 1
        l <- length(vec)
        vec[(i+1):l] <- vec[i:(l-1)]
        vec[i] <- a
        i <- i+2
      }
    }
    n <- n-1
  }
  return(vec)
}
```

B.10 mutation

The function **mutation** composes a 2-note piece of music following the rules of mutation. You can specify the following input parameters:

- a, b: the two notes
- n: number of mutations
- Anfang: beginning of the music piece (first measure)

Output is a sequence of the two notes a and b of length $(n + 1)$·length(Anfang)
The function can be called in R, e.g., via

```
mutation("a", "b", 4, c("a", "b", "a", "b"))
```

The code of the function looks as follows:

```
mutation <- function(a, b, n, Anfang){
  vec <- Anfang
  l <- length(vec)
  while(n != 0){
    vec2 <- rep(0,l)
    x <- sample(3,1)
    if(x == 1){
      for(i in 1:l){
        if(vec[i] == a){
          vec2[i] <- b
        }
        else{
          vec2[i] <- a
        }
      }
    }
    else if(x == 2){
      for(i in 1:l){
        vec2[i] <- sample(c(a,b), 1)
      }
    }
    else{
      vec2 <- vec
      i <- sample(1:l, 1)
      vec2[i] <- "break"
    }

    vec <- c(vec, vec2)
    n <- n-1
  }
  return(vec)
}
```

B.11 plot_spectrum

The function **plot_spectrum** plots the frequency spectrum of sounds, saved in csv-files, as a periodogram. You can specify the following input parameters:

- file_path: directory of the csv-file
- instrument, note: instrument and note, needed for the caption of the plot

The function can be called in R, e.g., via

```
DATA_PATH = ".../Data/"    # Please adjust data path!
setwd(sprintf("%s/The_Musical_Signal/Spectrum/", DATA_PATH))
par(mfrow = c(2,1))
plot_spectrum('Clarinett/bcl_c4.csv', 'Clarinett', 'c4')
plot_spectrum('Flute/fl_C4.csv', 'Flute', 'c4')
```

The code of the function looks as follows:

```
plot_spectrum <- function(file_path, instrument, note){
  data <- read.csv(file_path, header = FALSE)
  plot(data[,1], type = 'l', xaxt = 'n', xlab = 'frequency (Hz)', ylab = 'magnitude',
  main = paste('Spectrum of', instrument, note))
  x <- seq(0, 4000, 500)
  l = length(data[,1])
  axis(1, at = seq(0, l, (l/(length(x)-1))), labels = x)
}
```

Chapter C

Lists of Exercises for Higher-Level Topics

General

In this document we list the exercises corresponding to higher-level features.

List of Higher-Level Features

1. pitch (including chroma, chords, harmony, and key),
2. volume (including loudness, energy, rests, vibrato, and amplitude modulation),
3. timbre (including MFCC, spectrogram, formants, transients, instrument recognition, emotions), and
4. duration (including rhythm, measure, onset detection, ASDR identification, structure identification, tempo recognition) as well as the topic
5. genre (including music recommendation and organization).

C.1 Pitch (including chroma, chords, harmony, and key)

C.2 Volume (including loudness, energy, rests, vibrato, and amplitude modulation)

C.3 Timbre (including MFCC, spectrogram, formants, transients, instrument recognition, emotions)

C.4 Duration (including rhythm, measure, onset detection, ASDR identification, structure identification, tempo recognition)

C.5 Genre (including music recommendation and organization)